A Prediction and Prevention Model of Requirements Change Driven by the Improvement of Project Teams with Case Studies

Yuqing Yan, Zhenhua Zhang*
School of Mathematics and Statistics, Guangdong University of Foreign Studies, Guangzhou 510006, China.

* Corresponding author. Tel.: +86 13697465049; email: 200211098@oamail.gdufs.edu.cn
Manuscript submitted October 15, 2018; accepted December 27, 2018.
doi: 10.17706/jsw.14.2.47-57

Abstract: Requirements change management is a vital part of software project management. The existing literature on requirements change focuses on its technical aspects and is less concerned with the combination of personnel and technology. Few studies have focused on predicting and preventing changes in requirements and finding change predictors from the sociotechnical viewpoint. In this study, we examined prediction and prevention mechanisms of requirements volatility based on the analysis of change causes by stressing both the human and technical roles. Two case studies (a questionnaire and an interview) were done to validate the study. An online questionnaire was used to quantitatively analyze and explain the significance of human factors, including the emotional characteristics of developers. The result of the interview was applied to qualitatively illustrate the necessity and importance of effectively constructing and managing project teams to decrease the requirements change rate and enhance the software success rate. The prediction and prevention model established in this study was validated by a model called "3P + 2C", described in the interview. The paper concludes by suggesting to researchers four practical issues around the proposed model for further study.

Key words: Change cause, emotional factor, interview, rediction, prevention, questionnaire, strategy.

1. Introduction

Software requirements always change over time in response to the dynamically evolving demands of clients, stakeholders, organizations, policies, and the market environment [1]. Much attention has been given to the measurement and management of requirements changes throughout the software lifecycle [1]-[3]. A variety of methods and approaches have been proposed in the literature, including the classification and identification of change causes [2], [4]-[6]; conceptual ontology models of the requirements change process [5]; the rate of additions, deletions, and modifications of a set of requirements in a specific time[1]; change impact analysis [1], [7]–[10]; specific methods to minimize the impact of requirements change[8]; strategies for managing changes[11]; and the agile method, which addresses direct communications and extensive collaboration between customers and delivery teams [6].

More sophisticated techniques for managing requirements changes include formal methods and tools [12]; tools to measure requirements quality features and to evaluate automated requirements tools [13]; quantitative measurement of the quality of requirements[10]; evaluation of requirements change impact prediction tools[9]; and recently, the application by Knauss *et al*. [3] of just-in-time approaches to manage

nonfunctional requirements in agile projects based on the viewpoint of knowledge management.

However, few studies have focused on change prediction and prevention. This is probably the main reason that requirements change is still a critical factor affecting the software success rate, which has lingered at approximately 29% for the last 25 years [14]. Another reason is the lack of empirical studies throughout the area of software engineering, which has tended to use subjective information instead of solid empirical data [15]. A third reason is that most treatments in the literature of requirements volatility or changes are biased toward technical means and pay less attention to sociotechnical aspects [6].

This study aimed to build a sociotechnical approach based on a cause–effect analysis of the efforts to improve project teams' abilities to predict and prevent requirements changes. This approach was supported by two case studies. One provided arguments for emphasizing the importance of human factors in software development and management processes, and another supported the validation of the change prediction and prevention model established in the study.

2. Acquisition of Empirical Data and Adoption of Analytical Methods

This research used two case studies: a questionnaire and an interview.

2.1. Questionnaire

The questionnaire was an online survey carried out from www.wjx.com during the period of June 11 through July 10, 2018. The purpose was to determine the influence of human factors on software firms in China's Guangdong Province. Forty multiple-choice questions were asked, and each provided a maximum of 20 different answer choices. There were 323 valid responders from different companies all over the province, especially around the Pearl River Delta region.

The results were used to determine the significance of human roles in software development and management. The survey data were processed and analyzed by the arithmetic mean: the number of participants divided by total responders.

2.2. Interview

The interview was done in late July 2018 in a reputable software development company, ZZSDC of Guangzhou, China, which had been formed in 2001. ZZSDC specializes in developing e-government affairs systems. Its qualifications include ISO 9001, Capability Maturity Model Integration, and government certification as a national high-tech enterprise of China. In the interview, we asked for information about how ZZSDC implements software project risk management, including requirements change management and project teams management.

The material gleaned from this interview was summarized and qualitatively analyzed in detail by induction.

3. Importance of the Human Factor in Software Development Management

3.1. The Critical Factor in All Causes of Requirements Change

The identification of change causes in requirements has been identified as a way to implement a defensive management strategy[10]. Fig. 1 shows a hierarchical identification model of change causes [16].

The first level contains six top-level elements, each of which is decomposed into more concrete subcauses. This process was continued until it reached the bottom subcauses (inner causes) which were explicit requirements problems such as redundancy, incompleteness, and semantic ambiguity [4]. Of all factors, the one of stakeholders was the most important, especially the project teams, as shown in Fig. 2.

3.2. Human Factors Affecting the Success Rate of Software

Managing requirements variability lies in managing people [6], [11], [14], [17]. The three main problems or constraints caused by people that could cause volatile requirements have been found to be11] 1) cognitive constraints and lack of experience, knowledge, or both; 2) difficulties in or barriers to communication; and 3)

emotional and relational problems or constraints. Emotion management has already caught the attention of the software development community. The 2015 CHAOS report of the Standish Group set "Emotion Maturity" as one of the ten CHAOS factors of success, being rated at 15 points (the highest value), the same significance as Executive Sponsorship, User Involvement, and Optimization [18]. More than 30 years ago, Frederick Brooks said [19] "the quality of the people on a project, and their organization and management, are much more important factors in the success than are the tools they use or the technical approaches they take." Here, "the quality of the people" could be interpreted as not only the experience and knowledge of the personnel, but also their emotional characteristics. However, in reality software companies place little emphasis on emotional problems [10]. Much more research is still required on managing the emotional problems of stakeholders, especially project team members. What emotional factors are important in software development and its management processes? A recent online survey (see Section 2.1) found the answers shown in Table 1.

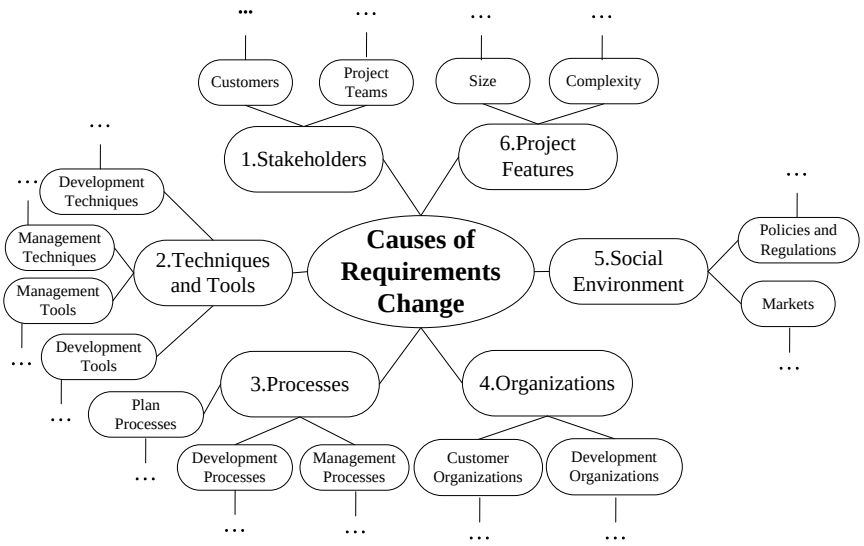

Fig. 1. Causes of requirements changes.

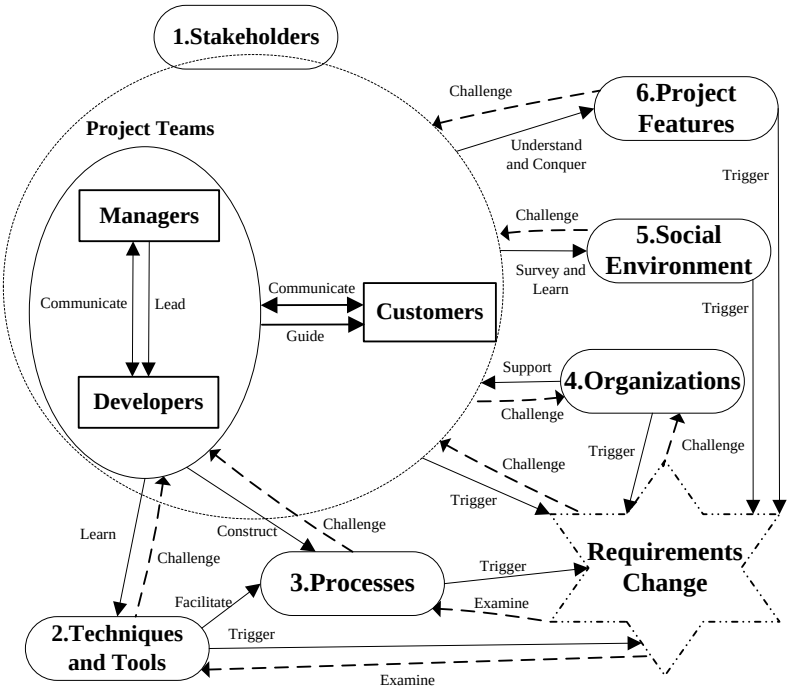

Fig. 2. Interaction of causes of requirements changes.

Table 1. Emotional Factors in SD and the Management Process[1]

Option	Pct.(%)[2]
Working with full enthusiasm	70.59
Good communication skills	68.11
Learning from outstanding people	64.40
Maintaining a stable mood	61.92
Responsible	60.99
Calmly and steadily handling things	59.44
Enjoy learning	54.49
Quickly plunging into work	40.87
Optimistic	39.63
Patient	37.15
Honest	27.86
Good temper	17.03

[1]From a questionnaire conducted in 2018 in Guangdong Province, China.
[2]Pct. (%) is an arithmetical mean: the number of responders divided by total responders (323).

Table 1 shows that “Working with full enthusiasm” got the highest percentage of responses. However, how can employee enthusiasm be stimulated? Maciaszek and Liong [20] thought that the project managers and leaders should be well selected and comprise those who “know how and when to manage and how and when to lead” and motivate people. They also said that “communication is the activity of exchanging information and knowledge.” For project team members, good communication skills were mandatory, which was confirmed by our survey.

Table 2 shows the human factors that affected software development success. It shows that "inexperienced developers" and "insufficient support and involvement from the customer organizations " were crucial factors that affected the success rate, which were truly coincided with the 2014 CHAOS report of the Standish Group [21].

Table 2. Human Factors Affecting the Success Rate of Software

Option	Pct.(%)
Project managers lacked leadership skills	18.27
Inexperienced project managers	28.48
Developers’ professional knowledge was incomplete	30.03
Unskilled developers	43.34
Inexperienced developers	48.30
Developers were not good problem-solvers	26.93
Lack of effective communication between team members	32.82
Lack of engagement, morale, and job satisfaction	21.36
Developers were not very responsible	31.58
Job roles change frequently	30.65
Employee turnover	34.30
Key employee turnover	43.34
Inadequate training	17.03
Lack of good communicate between developers and customers	32.51
Not enough customer involvement	25.39
Insufficient support and involvement from customer organizations	45.51

4. Establishing A Prediction and Prevention Model of Requirements Change Driven by Project Team Improvement

4.1. Related Works

Little research has been done on predicting and preventing requirements change. Loconsole and Börstler[22] proposed predicting requirements volatility by using the size of the requirements (number of lines in a requirements document) as the predictor. They alleged that their method for constructing prediction models could be applied to any software company. However, if requirements documents were not available or complete, their method could not be applied, and a single predictor could actually play a very minor role. Park et al.[23] explored the key requirements attributes from stories to predict possible changes (potential defects) in requirements specifications. They found two predictors: the number of indirect stakeholders and the number of related stories. It could be seen that [22], [23] shared the same predictor: the size of requirements.

Once a requirements change occurred, its propagation or ripple effect would be unexpectedly complicated [24], [25]. Assessing the effects was the task of change impact analysis based on the relations among requirements. There are two categories of relations: 1) A semantic relation built by linguistics [26] or nonclassical logic (such as predicate logic, belief logic, default logic, and paraconsistent logic) [27], [28]; and 2) Trace relations defined by business logic or development methodologies, normally detected by tools [29]–[32]. One task of the impact analysis of requirements change is to predict what other requirements or specifications would be affected by the changing requirements [32]–[35]. One common method is to stress the technical level but not the sociotechnical. That focuses on using explicit knowledge (e.g., requirements documents) instead of tacit knowledge (e.g., experiences of developers). Very little research discusses the control of requirements volatility from the view of social relations except that of Nakatani and Tsumaki [36]. They presented a method of predicting requirements volatility by analyzing the social relations among stakeholders (e.g., executives, competitors, and cooperative organizations), believing that changes in those relations would cause the stakeholders to change their requirements. Janes *et al*. [10] focused on defensive and reactive strategies by an empirical study. To implement a defensive strategy, they focused on identifying the causes of RC including the factors of human cognitive constraints, communication skills, and emotional problems. In fact, finding change causes was a method of predicting upcoming requirements changes in advance. This would identify harbingers of problems in requirements elicitation and analysis, which would help managers to adopt appropriate prevention and management measures to reduce the number of potential changes and make requirements more stable and complete. However, even if the prevention and management of changes was done strictly and seemed complete, requirements changes could not be avoided; hence, a reactive strategy should be used.

To implement a reactive strategy, Janes et al. introduced two methods: 1) Realizing flexible software solutions such as improving systemic learning, strengthening teamwork and communication, and encouraging proactive thinking; and 2) being flexible in the development process, such as using incremental delivery or an iterative development method.

4.2. Establishing the Model

When discussing the prediction of requirements volatility, multifaceted factors should be taken into account, especially the quality of stakeholders and project teams. However, the approaches cited in Section 1 took only a narrow view to produce predictor(s) by technical means. This study tried to improve the present situation, considering change causes by addressing the key role of stakeholders and project teams, and at the same time using change history documents (a technical part) to build a systematic model to predict and prevent requirements change. Fig. 3 is a schematic of the model DPPMRC for predicting and preventing requirements change by emphasizing the improvement of project teams.

As shown in Fig. 3, DPPMRC was built from a sociotechnical view, using both defensive and reactive strategies to manage requirements.

5. An Interview for Validating dppmrc

For validating the main idea of this paper, the model DPPMRC, an interview was done with the software company ZZSDC (see Section 2.2) , whose software success rate has been over 90% in recent years, owing to the implementation of good team management and corporate culture.

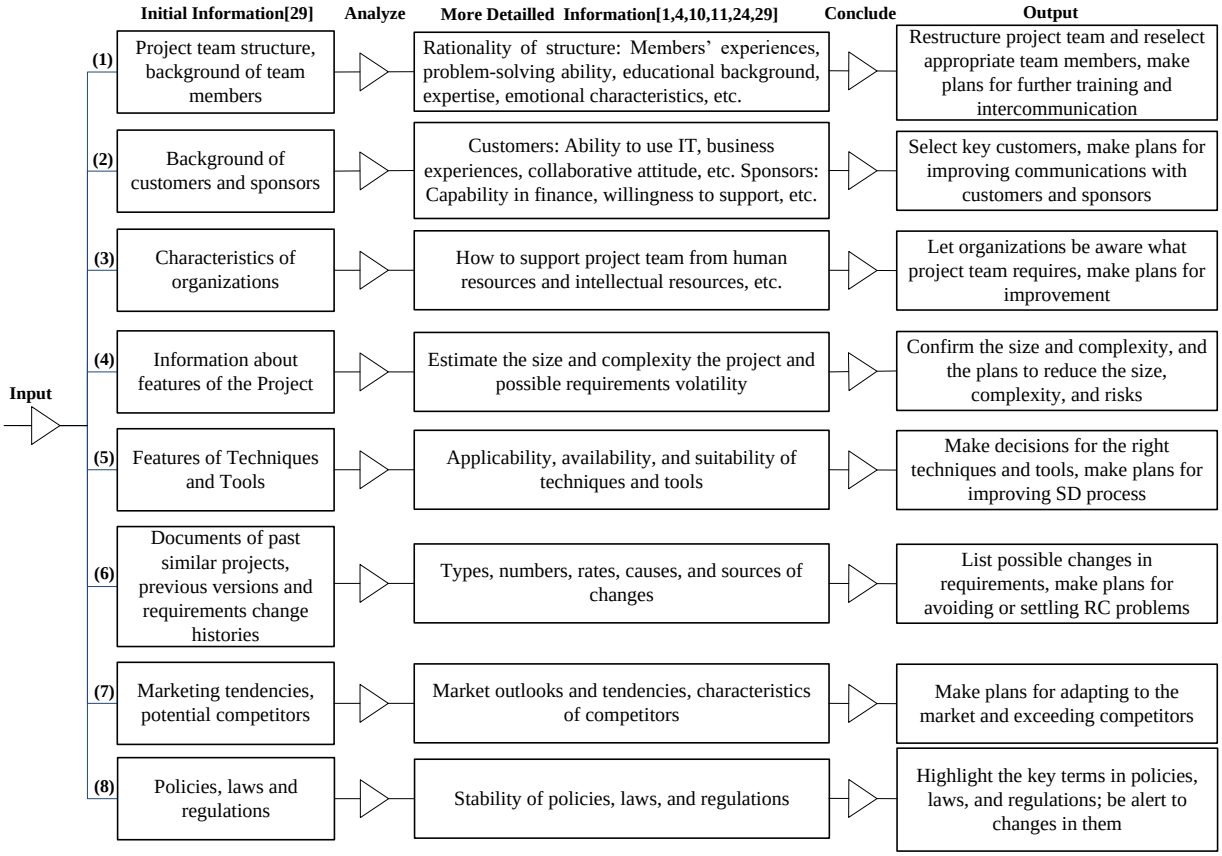

Fig. 3. Developer-centric prediction and prevention model of requirements changes(DPPMRC).

5.1. 3P + 2C Model: An Organizational Structure of Human Resources for Projects

The leaders of ZZSDC realized very clearly that people were the drivers for successful software development [14]. They said:"Software development company should learn how to organize people to make software well developed. From beginning to end, we treat every project as a management unit. By using a method called 3P + 2C, a project organizational management structure, to effectively organize people who only apply mature technology in development to make software delivered on time and on budget."

The 3P + 2C organizational architecture contained five different roles: product manager, project manager, program manager, customer manager, and consultant manager. Each project team was created using this pattern, and was made up of five subteams from four departments (Fig. 4).

In ZZSDC, a software project lifecycle (SPL) was normally separated into four phases: negotiation for obtaining a project (NOP); requirements acquisition and analysis (RAA); software design and coding (SDC); and software implementation, testing, and maintenance (SITM).

1) NOP. This was a marketing process led by the marketing and consulting departments. They were responsible for gathering marketing intelligence and building sales channels of customers in government departments; introducing, recommending, and establishing the good reputation of ZZSDC for potential government customers and consulting with representatives of different sectors of the government, such as the finance and tax bureaus.
2) RAA. When a project was agreed upon after the efforts of the marketing personnel and consultants, the product manager and corresponding engineers would play their roles. This was the phase of requirements elicitation, analysis, and design—crucial in the SPL. The product manager and requirements engineers acted as business experts with some knowledge of software engineering and SD. Most of the time they had to communicate with key customers to know what they wanted and how they normally carried out their activities, to learn whatever domain knowledge they may be unfamiliar with,

including laws and policies and the organization structures of the government department. Before SDC, they should know and understand very clearly all business procedures. They should also be able to very precisely extract the essential business aspects and accurately express user requirements using natural language and flowcharts (normally made by tools). This was a very important step in the SPL. If the requirements were not acquired correctly and completely, RC would occur in later phases, rework could not be avoided, and costs would increase greatly.

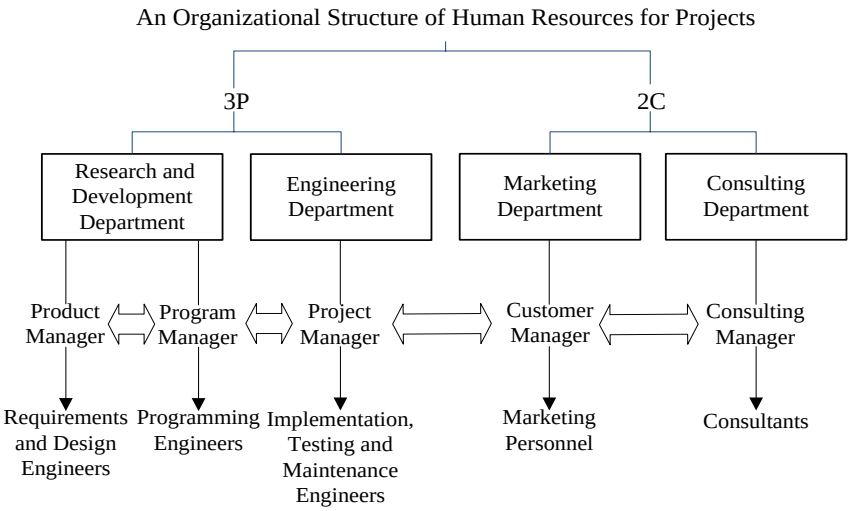

Fig. 4. 3P + 2C model.

3) SDC. The program manager also had to communicate with customers, although not as frequently as the product manager did, but was required to have more communication with the product manager to help create better designs. If the designs were not reasonable, the system would be wrong no matter how advanced the techniques were, would not satisfy users, and would fail. The program manager and the programmers were encouraged to use appropriate programming languages and techniques, as well as system architecture.
4) SITM. The project manager was also called the implementation manager. The software application was developed by iterative and incremental methods, but the implementation manager led the engineers to install, debug, test, and deploy the application. This work was user-oriented. While the system was being implemented, the users were being trained how to use the new or updated system. At the same time, user opinions were collected about usage experiences, interface friendliness, and system response time. The opinions were then submitted to the 3P departments as improvement suggestions for the next release.

The advantages of the 3P + 2C model had three aspects: 1) Subdivided roles and clear responsibilities. In this model, software engineers were divided into requirements engineers, design engineers, programming engineers, testing engineers, implementing engineers, and so forth. Each role had its own specific responsibilities, and the engineers knew clearly what those were. 2) Valuing customer relationships. First, 2C managers were customer-oriented; their duties were to communicate with potential users and to know about their markets and policies. Second, 3P managers also had to communicate with users, especially the project manager, who was in charge of installing, testing, debugging, deploying, and maintaining the software and training the users. 3) Emphasizing requirements acquisition. This model paid much attention to building connections with users, which was effective for eliciting further business requirements. However, more than that, the product manager and the team members were not purely technically oriented. They also had some knowledge about SD and computer science, and they were more like domain experts. They knew the users' business so well so that they could accurately describe user requirements by natural language and flowcharts and could design interfaces and interactive modes that would be provided to programmers. Their responsibilities were very crucial for the later

phases of SD and the success of software that could satisfy users. To become domain experts they had to learn. When a new project started for a new domain, such as a system of safety and supervision for a government financial department, they had to learn and study in depth the pertinent policies and laws of the government, the organizational architecture of the department's safety and supervision department, the methods of carrying out the business, and so forth. These responsibilities were very important for decreasing and preventing RC, enhancing the stability and completeness of requirements, and guiding software projects to success.

Also, the high success rates in ZZSDC were not just a matter of luck but were the outcome of a well-managed process. ZZSDC advocated 1) using mature technologies in SD that the employees had thoroughly mastered; 2) focusing on building the enterprise culture and regular project management as well as knowledge management; and 3) paying attention to humanistic care, the staff's sense of belonging, and personnel training, which could help to prevent the crises of experienced-staff turnover.

By the way 3P + 2C was implemented, it could be seen that the model was a distinct implementation of DPPMRC, except for the organization and project features parts. Those parts were carried out implicitly, however, because for the 17 years since 2001, ZZSDC did only one thing: develop e government systems for many provinces in China. The executives were highly educated in computer science and very experienced in SD, so they could easily assess the size and complexity of the projects and knew their potential competitors very well.

Because 3P + 2C could be considered a prototype of DPPMRC, a consequence of this interview is that it demonstrated that DPPMRC is effective.

5.2. Getting Executive Support

Another important element in ZZSDC's success in SD was its strong efforts to get executive support from user organizations. This effort was especially necessary in China, where nepotism and bureaucracy were still prevalent. Executive support was important in ensuring that the final product would meet expectations.

6. Significance

This study discovered that the current method of analyzing requirements changes underemphasizes the human role, consideration of which could be beneficial for establishing the prediction and prevention mechanism of volatility and changes in requirements. After a theoretical analysis guided by a sociotechnical view and studying two strategies of managing requirements change and two case studies, we built and validated the systematic model DPPMRC for predicting and preventing changes. This study could help researchers to find critical areas of comprehensive and practical prediction and prevention of requirements changes. This may also assist in creating a new methodological theory on requirements change management based on socio-technological theory.

7. Conclusion and Future Studies

This paper shares the results of two case studies for supporting the arguments and main subject of this study: establishing a systematic approach to actively manage requirements changes based on emphasizing the skills, abilities, and technical knowledge of personnel.

Future work would entail: First, continuing to test the effectiveness and availability of PPMRC by using it in a real software development environment; second, applying the main idea of PPMRC to improve software companies' project risk management and project team management; third, extending and tailoring it to make an enhanced version; and finally, either turning it into a practical software tool, integrating it into current requirements management tools, or both.

Acknowledgment

This work is supported in part by grants from the Municipal Philosophy and Social Science of The 13th Five-Year Plan Project of Guangzhou (2017GZYB45) and the Provincial Philosophy and Social Science of The 13th Five-Year Plan Project of Guangdong (GD17CGL18).

References

[1] Zowghi, D., & Nurmuliani, N. (2002). A study of the impact of requirements volatility on software project performance. *Proceedings of the Ninth Asia Pacific Software Engineering Conference*.

[2] Harker, S. D. P., Eason, K. D., & Dobson, J. E. (1993). The change and evolution of requirements as a challenge to the practice of software engineering. *Proceedings of the IEEE International Symposium on Requirements Engineering*.

[3] Knauss, E., Liebel, G., Schneider, K., *et al.* (2017). Quality requirements in agile as a knowledge management problem: More than just-in-time. *Proceedings of the International Requirements Engineering Conference Workshops*. IEEE Computer Society.

[4] Yan, Y. Q. (2018). Knowledge factors and models in requirements change management process based on causality. *Journal of Software*, *13(7)*, 386-394.

[5] Bano, M., Imtiaz, S., Ikram, N. *et al*. (2012). Causes of requirement change — A systematic literature review. *Proceedings of the International Conference on Evaluation & Assessment in Software Engineering*.

[6] Anitha, P. C., Savio, D., & Mani, V. S. (2013). Managing requirements volatility while "Scrumming" within the V-model. *Proceedings of the IEEE Third International Workshop on Empirical Requirements Engineering*.

[7] Goknil, A., Kurtev, I., & Berg, K. V. D. (2016). A rule-based change impact analysis approach in software architecture for requirements changes. 10-43.

[8] Ahmad, Z., Hussain, M., Rehman, A. *et al.* (2015). Impact minimization of requirements change in software project through requirements classification. *Proceedings of the ACM, International Conference on Ubiquitous Information Management and Communication*.

[9] Goknil, A., Domburg, R. V., Kurtev, I. *et al*. (2014). Experimental evaluation of a tool for change impact prediction in requirements models: Design, results, and lessons learned. *Proceedings of the Model-Driven Requirements Engineering Workshop*.

[10] Mohd, H., Mohd. R. B., & Sheikh, F. A. (2013). Overview of impact of requirement metrics in software development environment. *International Journal of Advanced Research in Computer Engineering and Technology*, *2(5)*, 1811-1815.

[11] Janes, A., Remencius, T., Sillitti, A. *et al*. (2013). Managing changes in requirements: an empirical investigation. *Journal of Software Maintenance and Evolution Research and Practice*, *25(12)*,1273-1283.

[12] Ghose, A. K. (2000). Formal tools for managing inconsistency and change in RE. *Proceedings of the Tenth International Workshop on Software Specification and Design*.

[13] Mohammad, U. B., & Shams, T. S. (2011). Metrics for requirements engineering and automated requirements tools. *Proceedings of the 5th National Conference*.

[14] Alan, Z. (2016). What we really know about successful projects. Retrieved from: https://www.scrumalliance.org/community/articles/2016/october/what we really know about successful projects#sthash.B1lUy96n.dpuf

[15] Jones, C. (2014). Software industry goals for the years 2014 through 2018. *Journal of Cost Analysis & Parametrics*, *7(1)*, 41-47.

[16] Yan, Y. Q., & Zhang, Z. H. (2016). A method to connect the source of software risk and requirements change[C]. *Proceedings of the 2016 International Conference on Management, Economics and Social Development*.

[17] Maciaszek, L. A, & Liong, B. L. (2004). Practical software engineering: A case study approach.

[18] Shane, H., & Stéphane, W. Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch[EB/OL]. Retrieved from: https://www.infoq.com/articles/standish-chaos-2015.

[19] Frederick, P. B. (1995). The mythical man-month: Essays on software engineering anniversary edition. Addison-Wesley Professional.

[20] Mcconnell S. (2006). Software estimation: Demystifying the black art.
[21] 2014 project smart. Retrieved from: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf
[22] Loconsole, A. (2008). A correlational study on four measures of requirements volatility. *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. British Computer Society*.
[23] Park, S., Maurer, F., Eberlein, A *et al*. (2010). Requirements attributes to predict requirements related defects. *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*.
[24] Yan, Y. Q., Li, S. X., & Liu, X. M. (2009). Quantitative analysis for requirements evolution. *Proceedings of the 2009 International Asia Conference on Informatics in Control, Automation and Robotics*.
[25] Yu, Q. Y., Li, S. X., & Sun, W. J. (2012). Dependency based technique for identifying the ripple effect of requirements evolution. *Journal of Software*, *7(3)*, 544-550.
[26] Morkos, B., Mathieson, J., & Summers, J. D. (2014). Comparative analysis of requirements change prediction models: Manual, linguistic, and neural network. *Research in Engineering Design*, *25(2)*,139-156.
[27] Didar, Z., & Ray, O. A. (1997). Logical framework for modeling and reasoning about the evolution of requirements. *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, IEEE Computer Society Washington*.
[28] Aditya, K. G. (1999). A formal basis for consistency, evolution and rationale management in requirements engineering. *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*.
[29] Yan, Y. Q. (2011). The study of requirements evolution and dependency. Guangzhou: Sun Yat-sen University.
[30] Antje, V. K. (2001). A trace model for system requirements changes on embedded systems. *Proceedings of the 4th International Workshop on Principles of Software Evolution*.
[31] Antje, V. K. (2002). Change-oriented requirements traceability. *Proceedings of the 8th IEEE International Conference on Software Maintenance, Support for Evolution of Embedded Systems.*
[32] Goknil, A., Domburg, R. V., Kurtev, I. *et al*. (2014). Experimental evaluation of a tool for change impact prediction in requirements models: Design, results, and lessons learned. *Proceedings of the Model-Driven Requirements Engineering Workshop*.
[33] Per, J., & Mikael, L. (2005). Impact analysis. *Engineering and Managing Software Requirements*. 117-142.
[34] Mikael, L. (1997). An empirical study of requirements-driven impact analysis in object-oriented software evolution. Linköping University, Institute of Technology, Sweden.
[35] Pedrycz, W., Iljazi, J., Sillitti, A. *et al*. (2016). Predicting the fate of requirements in embedded domains. *Proceedings of 4th International Conference in Software Engineering for Defence Applications*.
[36] Takako, N., & Toshihiko, T. (2014). Predicting requirements changes by focusing on the social relations. *Proceedings of the Tenth Asia-Pacific Conference on Conceptual Modeling*.

Yuqing Yan received her bachelor degree in mathematics from South China Normal University in 1985, the master degree and PhD in computer software and theory from Sun Yatsen University, Guangzhou, China in 1999 and 2011. She is an associate professor of Guangdong University of Foreign Studies, Guangzhou, China. As a researcher, she had been invited to do NIH project by Dr. Fu of Computer Science Department, University of Central Oklahoma, USA, from Oct., 2012 to July, 2013. She has published more than 20 scientific articles in the field of software requirements engineering management, five of them are: "Knowledge factors and models in requirements change management process based on causality," *Journal of Software*, vol. 13, no. 7, pp. 386-394, 2018; "An

ontology framework of requirements change management process based on causality," in *Proc. the ICPS published by ACM*; (3) "A method to connect the sources of software risk and requirements change," in *Proc. the ICMESD*, pp. 1179-1185; (4) "A risk-oriented cause identification framework of software requirements changes and case studies," *Software Engineering*, vol. 19, no. 12, pp. 5-9, 2016; (5) "Dependency based technique for identifying the ripple effect of requirements evolution," *Journal of Software*, vol. 7, no. 3, pp. 544-550, 2012. Her previous research interest was machine learning, artificial intelligence. Current research interest is in software requirements change management and knowledge management.

Zhenhua Zhang received his bachelor degree in mathematics from Hunan Normal University in 1998, the master degree in system science from Kunming University of Science and Technology in 2002, and PhD in computer science and technology from Nanjing University of Science and Technology in 2012, China. He is an associate professor of Guangdong University of Foreign Studies, Guangzhou, China. As a visiting scholar and researcher, he respectively visited Coventry Business School, Coventry University, and Department of Informatics, University of Leicester, UK, in 2014 and 2018. So far, he has published more than 50 scientific articles in the field of fussy reasoning, decision making, pattern recognition, data mining, risk management and software engineering. Five main articles are: 1) "A dynamic interval-valued intuitionistic fuzzy sets applied to pattern recognition," *Mathematical Problems in Engineering*, vol. 1, pp. 1-16, 2013. 2) "A construction approach of the distance measure with parameter between intuitionistic fuzzy sets based on the coordinates transformation," *Journal of Nanjing University*, vol. 53, no. 3, pp. 462-475, 2017. 3) "Deep extreme feature extraction: New MVA method for searching particles in high energy physics," *Filomat*, vol. 32, no. 5, pp. 1-15, 2018. 4) "A dynamic fuzzy group decision making method for supplier selection," *Journal of Applied Sciences*, vol. 13, no. 14, pp. 2788-2794, 2013. 5) "A study of tropical cyclone combination forecast model based on the cost-sensitive analysis," *Journal of Applied Sciences*, vol. 13, no. 22, pp. 5085-5091, 2013.