SURVEY REPORT USING EXTREME PROGRAMMING - DEVOPS IN MICROSERVICE ARCHITECTURE

Nagalambika[1]*, L. Manjunath Rao[2]

[1]Research Scholar, Department of MCA, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka 560056, India*

[2]Professor, Department of MCA, Dr. Ambedkar Institute of Technology, Bangalore, Karnataka 560056, India

ABSTRACT

The convergence of Microservice architecture, DevOps, and Extreme Programming in agile methodology has drawn massive attention from both academia and industry. The software industry is shifting towards agile methodologies and practices. This paper explores a survey report on the innovative software development process using Extreme Programming, DevOps, and Microservices to handle large, complex, and distributed systems. Successful agile implementation needs an efficient transformation model. We have conducted a survey to identify whether extreme programming and DevOps practices can successfully coexist with microservice architecture and their effects on development speed, component reusability, and overall architecture. Fifty people from diverse IT organizations have been interviewed, hypotheses have been designed, and results have been concluded based on the survey. The participants have also given recommendations that can help researchers work on certain factors.

Keywords: *Microservice architecture; Devops; Extreme programming; Agile methodology; Distributed systems; Component reusability.*

1. INTRODUCTION

Extreme Programming (XP) is an agile methodology that emphasizes principles like test-driven development, continuous integration, and frequent short releases. When XP is combined with Microservice architecture and DevOps practices, it offers a strong framework for developing and maintaining large, complex, and geographically distributed applications. It enhances software development speed, agility, and quality [1], [2]. This paper delves into a survey-based exploration of an innovative software development process that leverages the synergies between Extreme Programming, DevOps, and Microservices to address the challenges posed by large, complex, and distributed systems. Microservice architecture emerges from real-world architectural patterns. Using lightweight communicating mechanism services are collaborated to meet their goals. Netflix, Amazon and Uber have adopted microservices architecture. This approach is an independent way of creating applications that can interoperate and integrate efficiently to promote low coupling and high cohesion. Maintenance efforts can be minimized by avoiding the effect of changes from one microservice to another. Services should be deployed distinctly to enhance independence and design maintainability. Maintenance is either stable or adaptive depending on functional requirements. This architectural style is based on developing a package of small services in a single application, where each service is running independently [3]. Due to this researcher are trying their best to provide adequate support to enterprises leading to the quick rapid development of heterogeneous tools and solutions for microservices implementation.

1.1 Problem Statement

Extreme Programming, a well-known agile methodology, is unable to manage complex and geographically distributed applications necessitate exploration of alternative approaches. Integrating XP with Microservice architecture and DevOps practices presents an opportunity to transform the software development process. Microservices help break down large applications into small-scope but independently functioning services to provide agile service plus flexibility for developers, but this does not come without challenges. A microservice implements functionality over a network using lightweight protocols [4]. Concurrently, Microservice architecture and DevOps practices have gained traction for their ability to enhance agility, scalability, and resilience in software systems [2], [5]. Microservices helped with faster development and change cycles. Additionally, hypothesis testing has been conducted against each objective outlined in the study.

1.2 Purpose of Study

The purpose of this survey study includes the following objectives:

- Development of innovative software development process using extreme programming, DevOps, and microservice architecture to execute medium, large-scale, and complex projects.
- Use of proposed process in extreme programming to provide support to execute geographically distributed projects, if programmers are located geographically.
- The use of the proposed process for both code and design-centric development.
- The proposed process provides an overall design of the system and provides necessary documentation.
- The proposed process supports the reusability of existing components.

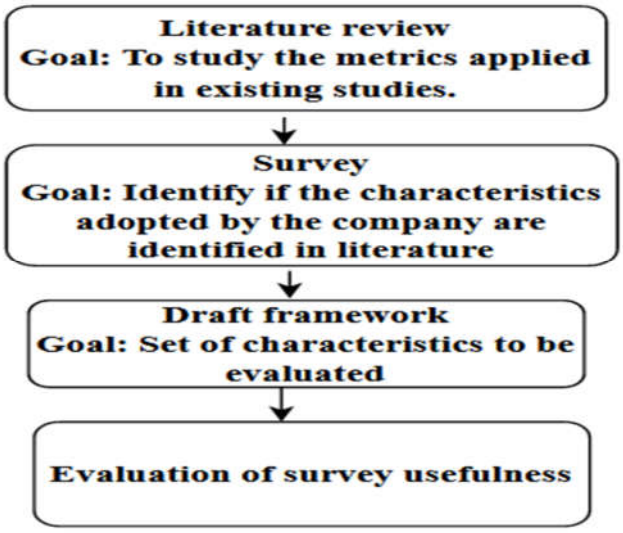

Figure.1: Steps involved in Survey

1.3 Research Questions

From the above-mentioned objectives, the resulting research questions are mentioned as:

- Is it possible to develop a software development process that uses microservice architecture in extreme programming to execute medium, large-scale and complex projects?
- Does the proposed process support to execute geographically distributed projects, if programmers are located geographically?
- Does the proposed process provide an overall design of the system and provide the necessary documentation?
- In contrary to XP being code centric, can it be used for design centric development?
- Does the proposed process support component reusability?

1.4 Formulation of Research Hypothesis

Based on the above research question we have formulated seven hypotheses as follows:

1.4.1. Hypothesis 1

Ho: There exist no significant relations between using XP and microservice architecture to execute large, complex and distributed applications.

H1: There exists a significant relationship between XP and microservice architecture to execute large, complex and distributed applications.

1.4.2. Hypothesis 2

Ho: The proposed XP practices are necessary for distributed applications.

H1: The proposed XP practices are not necessary for distributed applications.

1.4.3. Hypothesis 3

Ho: The proposed process does not support Full Development Plan and Release Plan. Hence XP is still code-centric, not design-centric development.

H1: The proposed process supports Full Development Plan and Release Plan. Hence XP is still code-centric, not design-centric development.

1.4.4. Hypothesis 4

Ho: The proposed process does not provide overall design of the system and provide necessary documentation.

H1: The proposed process provides overall design of the system and provides necessary documentation.

1.4.5. Hypothesis 5

Ho: The proposed process does not support component reusability.

H1: The proposed process support component reusability.

2. REVIEW OF LITERATURE

In this paper, the authors have described microservices-based NFV architecture. Not only benefits and challenges but also the results are emulated under KPIs. The complexities of microservices in telecom networks have been analysed along with a discussion of AI in complex decisions. Authors have proposed that artificial intelligence can be used to decide aggregation and disintegration. Authors have also investigated the benefits and challenges of this new approach through analysis and actual testing by providing results for the performance of the microservices approach under some KPIs and set the contacts for applicability of the same in NFV [4].

In this paper, the authors have introduced the idea of variability in microservices. The progress examples of microservices and SPLs are the same in terms of facilitating reuse and customization. In terms of functionality six challenges are presented that relate SPL with microservices. Authors have intended to get the best of both by combining [6]. While delivering commercial-grade software time and place plays an important role. This approach has worked on independent tools. The complex test set of independent tools helped in reducing the delivery time. But this method may not be upright for all demeanours of software engineering [7].

Although cloud and mobile computing are shaping the applications with their unique features like scalability and fault tolerance. But the challenges like security and reliability cannot be overseen. Behaviour-driven development has been popular for the last few years. It got its popularity from its executable acceptance tests (EAT). These tests describe the expected behaviour and acceptance criteria of features using simple readable syntax. While applying the BDD framework reusability, maintainability and auditability are the major areas of concern. A reusable but automated testing architecture presented to report all these questions. BDD acceptance tests

can be applied across multiple repositories leading to reducing the struggle of developers and testers. They are allowed to independently iterate on separate repositories [8].

Authors have deeply studied containerizations and component adoption in monolithic systems. To understand common patterns in component selection set of open-source projects are identified in terms of functional size to be shared among different teams. This helped increase the fitness and quality of collected data [9]. During XP2018 the first set of interviews was conducted, while in [9] goals, methodology, problems, advantages and challenges of microservices applied in an agile development process are discussed. The results of a XP2017 workshop are analysed. The adoption of DevOps principles promotes seamless collaboration between development and operations teams, leading to enhanced deployment frequency and reliability [10], [11]. Furthermore, geographical dispersion of development teams poses additional coordination and communication challenges, which need to be addressed for successful project execution [1], [12].

Authors in [13] have proposed a framework to automatically integrate microservices into the built system and also provide dynamic scalability on multiple platforms. Distribution scalability is handled by the server between clients. Chosen parameters are combination, execution time and the number of calls. The future direction of this research is to the successful integration of the system into a website or API. This paper reports a survey examine agile requirements engineering (RE). It is also discussed that if requirements engineering can remedy the challenges of traditional requirement engineering and the new agile challenges [14]. Based on above literature, a questionnaire-based survey is designed. The respondents are 50 practitioners from a large software development company. The experts shows that agile practices address some requirement engineering challenges. Authors in [15] have stated coherence as the critical stage for

distributed agile transformation. Authors have also stressed on provision of dedicated resources to support the transformation.

Authors have proposed a system where addition of microservices is easy and automatic. The benefit of the proposed approach lies in the easy scalability of the system [16]. In authors have worked on academic information system based on advanced personal extreme programming system. The development process is tested and four proposed features have produced valid results.

Authors in [17] provided insights into distributed systems, highlighting considerations for designing resilient and scalable architectures in distributed environments. In [18] presented a case study on the implementation of microservices architecture, offering insights into real-world challenges and solutions encountered during the adoption process. Authors have deeply studied advocated for an agile approach to software development using cloud computing technologies, emphasizing the benefits of scalability and flexibility offered by cloud-based platforms [19].

Author in [20] focus on testing challenges and solutions in microservices, shedding light on critical aspects of software quality assurance., while [21]-[23] explore dynamic microservices architecture and discuss edge and fog computing architectures. In [24]-[25] analyze microservices in industrial big data analytics, provide insights into architectural and operational aspects of microservices, while examine microservices for IoT applications in smart cities.

3. RESEARCH METODOLOGY

This section describes the statistical methods adopted and the final results. The respondents are from different designations, age groups and companies. A case study was carried out in an organization to identify XP practices, Continuous integration and Microservice architecture. The selected organizations develop and use extreme programming for large size and complex software.

A questionnaire was prepared to accesses the suitable practices for such large scale and complex software development. Online shopping application was used for this study, as companies were reluctant to give their data for this study due to privacy concerns and organizational intellectual property. The survey's objective is to examine the reasons for switching from monolithic to microservices-based architectures as well as its benefits and drawbacks. We organized the questions based on the data we needed to gather like company and personal information, role of interviewees and role of software. (As shown in Figure 1)

3.1 Population

Sample size for the research comprises a diverse group of professionals from various IT designations, including Managers, Team Leaders, Architects, IT Consultants, and Design Specialists. The sample is characterized by a range of experience levels, spanning from 2 to 22 years. This diversity ensures a comprehensive exploration of insights and perspectives within the dynamic landscape of the information technology sector.

3.2 Sampling Frame

Table 1. Shows the sampling frame for N=50

S. No	Designation	Experience in Years	Frequency
1	Manager (Delivery Manager, Senior Project Manager, Program Manager and QA Manager)	15-22	11
2	Team Leader	10-17	10
3	Architect	10	9
4	IT Consultant	2-17	5

5	Design Specialist	15-20	5
6	Academia	3-15	>10

3.3 Research Design

The descriptive, non-experimental method of the survey has been adopted. The respondents of the survey are working in various IT companies, and are avid users of microservices. The survey includes primary data collection using the mode of Questionnaire.

3.4 Research Instrument

This research consists of a questionnaire with closed-ended questions. It comprises seven key elements. These elements cover the overall objective of the study. For grading, we have used a 5-point Likert scale. This scale starts from 1 to 5 points; strongly agree receives 5 points, and strongly disagree receives 1 point. The information was gathered from a sample of 50 respondents who worked for private companies. The data were then analyzed using statistical methods like the mean, standard deviation, and t-test.

3.5 Data Analysis

To examine and assess the data, we have adopted following methods

1. Mean
2. Standard deviation
3. T-test to compare groups

For the study's analysis, Stratified Random Sampling, Simple Descriptive Analysis, and an Independent Sample t-Test are used (3). Statistical significance is defined as a p-value of 0.05 or less (typically ≤ 0.05). Given that there is a less than 5% chance that the null hypothesis is true (and that the data are random), this implies strong evidence against it. As a result, we accept the alternative hypothesis and reject the null hypothesis if the p-value falls below your level of

significance, which is commonly p 0.05. In total, this research comprised of five objectives. Data is collected by means of questionnaire.

The first objective of the research is to develop a software development process, to execute medium, large-scale, complex Projects using extreme programming and micro service architecture.

The following sections of this research paper shows the results of the designed hypothesis, altogether this research comprised of five objectives which are discussed in forthcoming sections.

Table 2: Shows the result of hypothesis with respective to the roles

Research Question	p-value			
	Architect	Developer	QA manager	Delivery manager
Suitability for large and distributed applications?	0.023	0.021	0.022	0.021
Support component reusability	0.021	0.022	0.024	0.221
Loosely coupled	0.016	0.011	0.013	0.012
Support over the globe	0.031	0.021	0.032	0.033
Improvement in development speed and efficiency	0.052	0.055	0.061	0.012
Improvement in quality of applications	0.031	0.032	0.023	0.022

XP practices implement in the Application	0.002	0.003	0.002	0.002

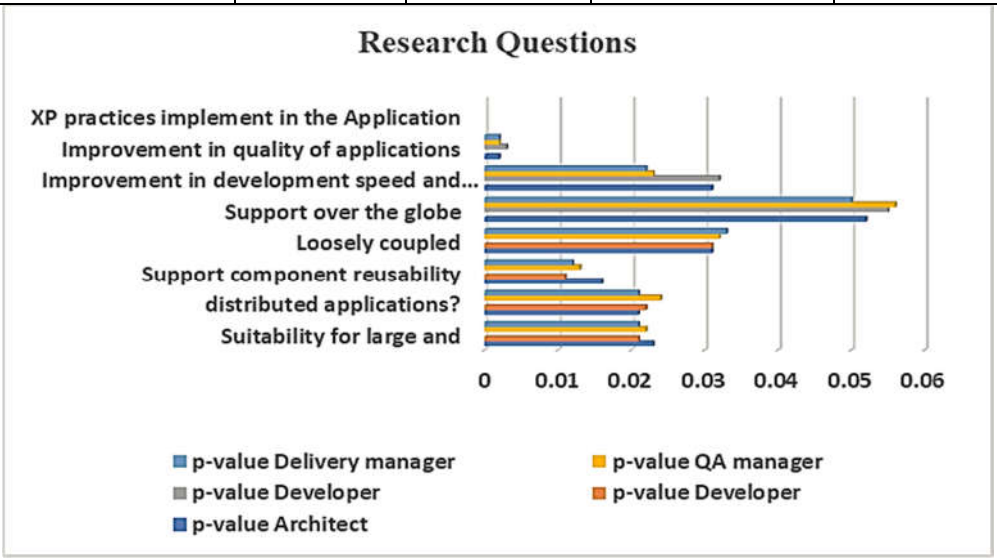

Figure 2: The graphical representation shows the result of hypothesis with respective to the roles

Table 3: Shows the type of XP practices that can be implemented in Microservices

XP Practices	p-value			
	Architect	Developer	QA manager	Delivery manager
Daily Stand-up meetings	0.022	0.031	0.031	0.034
Adaptive planning	0.021	0.021	0.033	0.023
Code Control	0.012	0.044	0.039	0.034
Continuous Integration	0.044	0.009	0.025	0.005
XP Project management	0.077	0.023	0.088	0.067
Visual Indicators	0.033	0.076	0.078	0.067
Code Gallery	0.008	0.008	0.021	0.012

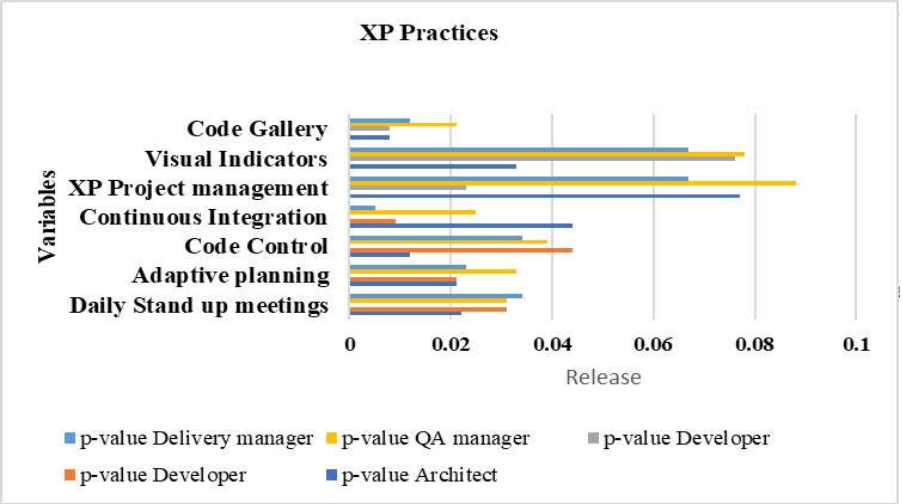

Figure 3: Graphical representation shows the type of XP practices that can be implemented in microservices

The results pertaining to the first objective are presented in Table 2 and Figure 2, showcasing the outcomes of hypothesis testing concerning various roles involved in the development process including "Architect," "Quality Assurance Manager," "Developer," and "Delivery Manager. Table 2 illustrates the statistical significance, represented by p-values, regarding the suitability of the proposed process for large and distributed applications, support for component reusability, loose coupling, global support, enhancement in development speed and efficiency, quality improvement of applications, and implementation of XP practices within the application. Similarly, Figure 2 provides a graphical depiction of these results across different roles involved in the project.

Additionally, Table 3 and Figure 3 provide detailed insights into the specific XP practices that can be successfully applied to facilitate the execution of medium to large-scale, complex projects within the framework of extreme programming and microservice architecture. These findings are pivotal for grasping the practical utility and potential ramifications of integrating XP methodologies into the realm of microservices development.

The second objective of this research paper is to investigate how the proposed process, with support from extreme programming, facilitates the execution of geographically distributed projects, if programmers are located geographically. The values are computed on the basis of release 1, 2, 3 and 4.

Table 4: Geographical execution of program

Geographically program execution	p-value			
	Release 1	Release 2	Release 3	Release 4
Code Exhibit	0.023	0.021	0.022	0.021
User stories completed	0.021	0.022	0.024	0.221
User stories Increase	0.016	0.011	0.013	0.012
Story Points	0.031	0.021	0.032	0.033

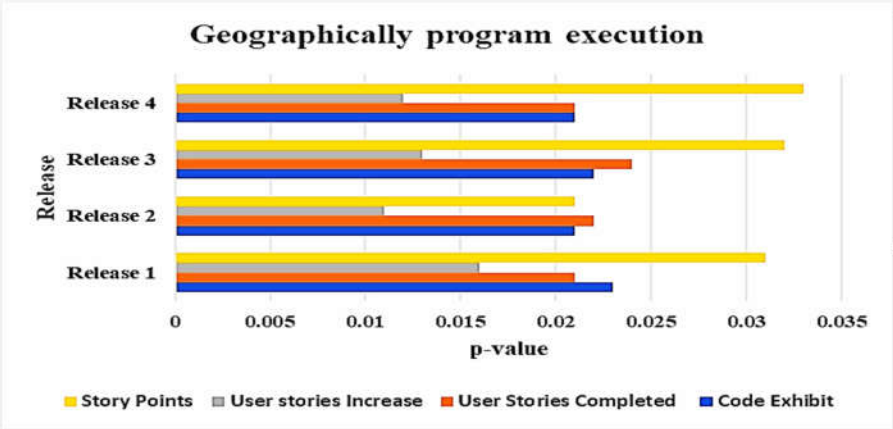

Figure 4: Geographically program execution

The table 3 and figure 4 show the result that for distributed applications, story points, increase in user stories, User stories completion and code exhibit are the necessary practices. The result from the table (2) supports this.

The third objective of this research is that by using proposed process, extreme programming can be used as both code and design centric development. For this we have considered the variables called microservice architecture, code reusability and new code. The results have shown that by implementing the proposed approach the extreme programming can be implemented as both code and design centric way.

Table 5. Code and Design Centric Development

XP Process	p-value			
	Architect	Developer	QA manager	Delivery manager
Microservice Architecture	0.031	0.022	0.024	0.221
Code Reusability and new code	0.016	0.011	0.013	0.012

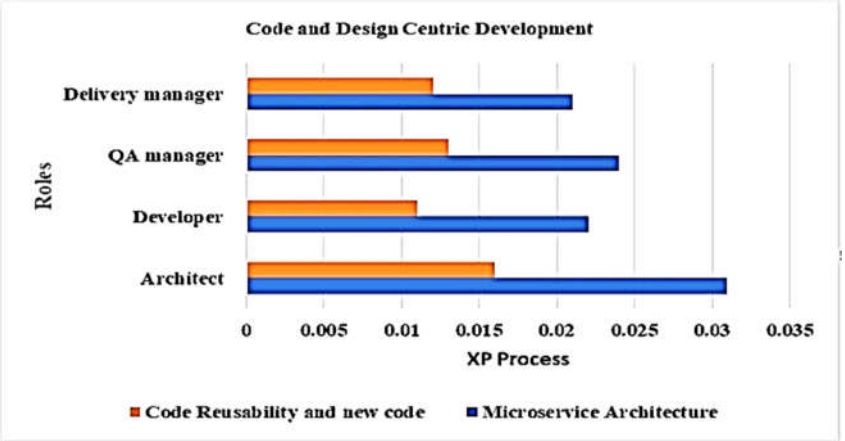

Figure 5: Code and Design centric Development

Figure 5, 6, 7 are given The fourth objective of this research is that the proposed processes provide overall design of the system and provide necessary documentation, as evidenced by Table 6's responses from a 21-day sprint showcasing performance metrics like training hours, iterations, architectural support, collaboration, satisfaction, interest levels, and velocity targets. These responses serve as evidence of the proposed processes' efficacy in delivering comprehensive system design and documentation and their alignment with different project roles, supported by p-values indicating statistical significance.

Table 6: Full Development Plan

S. No	Project Performance Measurements	Expected	Architect	Developer	QA manager	Delivery manager
1	Numbers of Hours Spent in Training and Up gradations.	40 hours -- (8 Hours Each day for 5 Days).	0.03	0.02	0.04	0.04

2	Number of Iterations for 4 releases	6 Iterations.	0.02	0.01	0.02	0.03
3	Application Architectural Design Change support	yes	0.03	0.01	0.02	0.03
4	Team Collaboration	High	0.005	0.006	0.003	0.003
5	Satisfaction (working environment/Culture)	High	0.02	0.01	0.02	0.03
6	Interest(buy-in)	High	0.03	0.02	0.03	0.04
7	Velocity Increase (Speed of deliverable	4%	0.03	0.03	0.04	0.04
8	Code Gallery	8.6 Interest(buy-in) High	0.02	0.03	0.03	0.03
9	Weekly work Hours	8.7 Velocity Increase (Speed of deliverable) 4%	0.03	0.03	0.02	0.02

The fifth objective is that the proposed process support reusability of existing components. The results of this have are shown in table (1). The results indicate that there is a significant connection between extreme programming and microservice, the lesser values of ‘p’ test indicate rejection of the null hypothesis. Table 2, shows that the value of ‘p’ is less than ‘0.05’, which

indicates that the null hypothesis is rejected. This means that the proposed XP practices are necessary for distributed applications. The proposed process also supports component reusability. The proposed process improves the development speed and cost-efficiency of software applications. The proposed process support loosely coupled and deployed across the globe. The proposed process affects the quality of large and distributed applications, proposed process supports the Full Development Plan and Release Plan. So, XP can be used as both code and design-centric development architecture to execute large, complex and distributed applications.

4. INFERENTIAL STATISTICS

The results indicate that there is a significant connection between extreme programming and microservice, the lesser values of ‘p’ test indicate rejection of the null hypothesis. Table 2, shows that the value of ‘p’ is less than ‘0.05’, which indicate that the null hypothesis is rejected. This means that the proposed XP practices are necessary for distributed applications. The proposed process also supports component reusability. The proposed process improves the development speed and cost-efficiency of software applications. The proposed process support loosely coupled and deployed across the globe. The proposed process affects the quality of large and distributed applications; proposed process supports the Full Development Plan and Release Plan. So, XP can be used as both code and design-centric development architecture to execute large, complex and distributed applications. The survey has also reported the issues faced while migration towards the distribution.

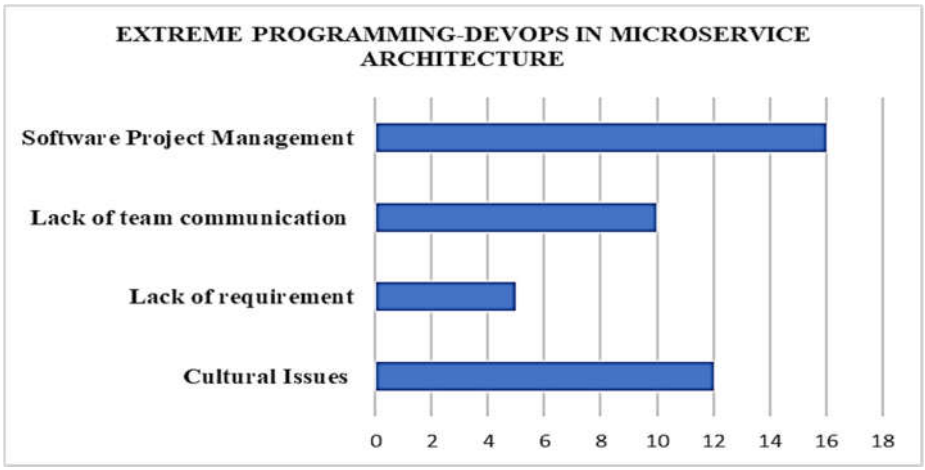

Figure 6: Issues faced while after agile adoption

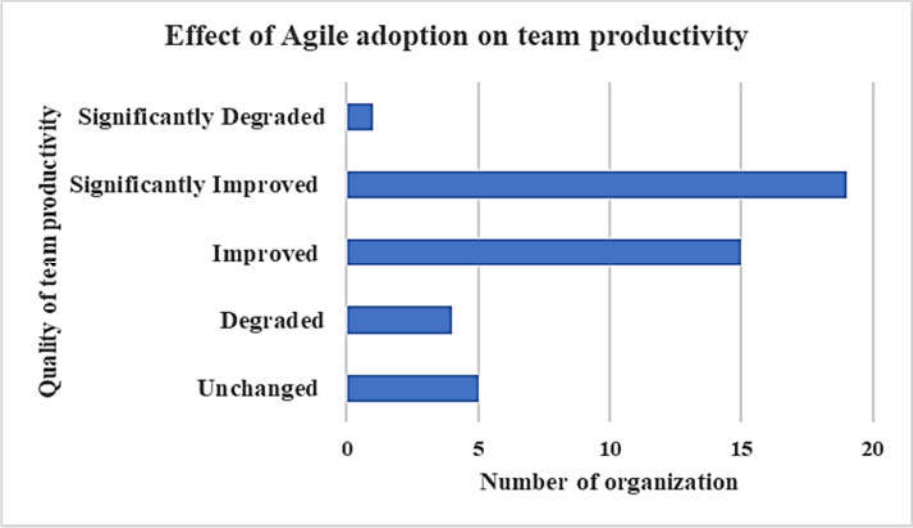

Figure 7: Effect of agile on productivity of the teams

5. SHORTCOMINGS OF THE APPROACH

1. The quality of results from hypothesis testing can be impacted by a number of factors. We had to rely on the stopping rules, which were open to numerous interpretations and comparisons, in order to evaluate the p value for our observation.

2. The estimation value is while focusing on statistical significance of the data we have ignored the estimation value while repeating the tests that sometime resulted in ambiguous test value.

3. Since it was challenging to find candidates with experience in the migration to microservices-based architectural styles, we did not pre-plan the sample of interviews.

4. To understand the costs and importance of migration from monolithic to distributed microservices. We have collected the information with the help of questionnaire that consist of closed and open-end questions. In order to score their responses, we requested that the participants use a five-point Likert scale, where 0 denoted "completely irrelevant" and 4 denoted "fundamental."

6. FINDINGS

- Everyone who took part consistently stated and ranked software maintenance as being of the utmost importance, which is one of the findings that led to the adoption of microservices-based architectures. Only migration experts mentioned additional factors for acceptance, such as scalability, assigning responsibilities to independent teams, and simple DevOps support.
- Monolithic systems can be made simpler by using the modular architecture of micro services. Distributed development is made simpler by breaking a system down into distinct, self-deployable services that allow development teams to make modifications and test their service independently of other developers. Additionally, the modest size of each microservice helps to make the code more understandable, which enhances its capacity to be maintained.
- It is simpler to scale microservices than it is to scale monoliths. Monolithic systems must be scaled with significant hardware investment and frequently with code alterations. If a particular component is the bottleneck, more potent hardware can be deployed, or many instances of a single monolithic program can run across various services and be controlled by a load balancer.

- A specific microservice that is the bottleneck in this scenario can be containerized and run across multiple hosts in parallel without relocating the entire system to a more powerful machine.

7. CONCLUSION

The proposed process can make the development faster, cost-efficient including loose coupling and scalability across the globe, and provide the necessary documentation. The documentation helps efficient coding based on skillset. This approach is both code and design-centric without affecting the quality of large and distributed application. The reusability component reduces the development effort time and provides quality software. It can handle disaster Management with the help of a data center by applying a load balancer. In future, we have planned to conduct interviews of industry professionals who are using and implementing microservices to better understand if and why practitioners follow some best practices and if they do not follow some practices what are the reason behind it. Our future work also includes an industrial-scale case study with companies that adopt microservices to monitor real software developers developing microservices-based projects to report the problems they face and the solutions they apply.

8. RECOMMENDATIONS AND SUGGESTIONS FROM THE STUDY

1. Using Microservice Architecture and DevOps combination we can ensure the quality of application due to in max phase of DevOps testing is carried out.
2. As concert to Microservice and Agile (XP) is best suited for the large and distributed project, due to independent development, deployment and loosely coupled features.

3. This process works very well with all kinds of projects.
4. 40 Hours Spent in Training is depending on the complexity of the requirements
5. Productive “Weekly work Hours” is less when considered some ad-hoc meetings/ Sick leaves/Employees leaving the organization. Etc., hence it is always considering some buffer hours during the planning.
6. Agile methodology is good for the organization and client where it will be more burdens to the developer and tester as they need to complete the sprint on time and for this, there is a chance to do mistakes in development and testing due to short of time and pressure.
7. This supports a larger kind of application and a well-suited design process.
8. Each sprint has 10 days, out of which 8 days for the development cycle, 1 day for sprint planning and 1 day for sprint review and retrospective.
9. For each day we consider only 6 hours for development work. And 2 hours for meetings and training.
10. Here the assumption is that everyone is available for the duration of the sprint and both planned and unplanned leaves are considered.
11. The user stories are clear to everyone with requirements are frozen.

ACKNOWLEDGMENT

I, sincerely thank all the participants from diverse IT organizations for their invaluable insights that shaped this survey. Special appreciation goes to research guide for their exceptional guidance and unwavering support, which were instrumental in shaping the outcomes of this study.

REFERENCES

[1] M. Fowler, "*Microservices: A definition of this new architectural term*," 2014. Retrieved from https://martinfowler.com/articles/microservices.html

[2] S. Newman, "*Building microservices*," in O'Reilly Media, Inc., 2021.

[3] Davide T, Valentina L, Claus P, and Andrea J, "Microservices in Agile Software Development: a Workshop Based Study into Issues, Advantages, and Disadvantages," *XP '17 Workshops,* Cologne, Germany, 2017, doi: 10.1145/3120459.3120483.

[4] B. Elizabeth, W. Krzysztof, and R. Bjorn," A case study on benefits and side-effects of agile practices in large-scale requirements engineering," *in proc of First workshop on Agile Requirements Engineering*, pp. 1-5, 2011, doi: 10.1145/2068783.2068786.

[5] R. C. Martin, "*Clean architecture: A craftsman's guide to software structure and design*," Prentice Hall, 2017.

[6] K.A. Harish, and B. J. Prabha," Scientific assessment and evaluation of ineffectual human phenomenon in acquisition design and development," *in pro of Materials Today*, 2020, doi: 10.1016/j.matpr.2020.12.1074.

[7] B. Mihai, I. Adrian, and G. Daniela, "Dynamic microservices to create scalable and fault tolerance Architecture," *in Proc Computer Science 159*, pp. 1035-1044, 2019, doi: 10.1016/j.procs.2019.09.271.

[8] H. Mike, J. Michael, Patrick Ct, Byron C, Jonathan PB, Tiziana M," Software engineering and formal methods," Communications of the ACM, Vol.51, No.9, pp.54–59, 2008.

[9] M. Nekovee et al., "Towards AI-enabled microservice architecture for network function virtualization," *in Proc of IEEE Eighth International Conference on Communications and Networking*, pp. 1-8, 2020.

[10] T. Woods and M. Lin, "Container orchestration and microservices: An analysis of Docker Swarm and Kubernetes, "*in IEEE 23rd International Enterprise Distributed Object Computing Conference*, pp. 54-61, 2019.

[11] W. Vogels, "Beyond ACID: A new paradigm for transaction processing," *ACM Queue*, vol. 4, no. 6, pp.48-55, 2006

[12] C. Yanaga, and M. Toum, "Microservices: The pitfalls," IEEE Software, vol.33, no.3, pp.122-124, 2006.

[13] M. Rahman and J. Gao, "A reusable automated acceptance testing architecture for microservices in behavior-driven development," *in Proc of IEEE Symposium on Service-Oriented System Engineering*, pp. 321-325, 2015.

[14] VO Rory, E. Peter, MC Paul," Continuous Software Engineering – A Microservices Architecture Perspective," *Journal of Software: Evolution and Process*, Vol.29, No. 11, 2017.

[15] L. Valentina, and S Outi, "Software components selection in microservices-based systems," *in Proc of Conference on Agile Software Development: Companion*, 2018.

[16] KGA Wesley, K. Jacob, DFM. Willian, "Variability Management meets Microservices: Six Challenges of Re-Engineering Microservice-Based Webshops," *Pervasive Health: Pervasive Computing Technologies for Healthcare*, pp.14–24, 2020.

[17] C.Saul, and M. Burrows, "*Distributed systems for fun and profit*," 2018. Retrieved from https://book.mixu.net/distsys/

[18] J. Pokorny, J. Vojtisek, and I. Kolingerova, "Microservices architecture implementation: A case study," *in Proc of the 7th International Conference on Information Management and Engineering*, ACM, pp. 153-157,2021.

[19] M. Smith and P. Trott, "Developing software as a service: An agile approach using cloud computing," *Addison-Wesley Professional*, 2017.

[20] R. Kazhamiakin and T. Männistö, "A survey of testing challenges and solutions in microservices," *Journal of Systems and Software*, 171, 110897, 2021.

[21] K. Kugler, U. Hohenstein, and W. Hasselbring, "Dynamic microservices architecture using function-as-a-service," *Future Generation Computer Systems*, 117, pp.181-193, 2021

[22] R. Polli, and Di Francesco Maesa, D., "*Microservices-based architectures for edge and fog computing: A survey*," Future Generation Computer Systems, 120, pp.503-527, 2021

[23] Y. Zou et al., "Microservice architecture and its application in industrial big data analytics," *Computers in Industry*, vol. 124, 103408, 2021

[24] R.F Medeiros et al., "Architectural and operational aspects of microservices: A systematic literature review," *Journal of Systems and Software*, 176, 110938, 2021

[25] H. Lee, et al., "Microservices-based architecture for IoT application in smart cities," Journal *of Network and Computer Applications*, Vol.182, 102973, 2021.

Author Biography

Nagalambika Swamy is a Research Scholar, At Visvesvaraya Technical university and currently working as Assistant Professor in the Department of Computer Science at Ramaiah College of Arts, Science, and Commerce, Bangalore, India. She has worked in various Software Industry and her Research areas are Software Engineering, Data Science, and Artificial Intelligence With a total of 16 years of experience in academia and industry. She can be contacted at email: nagalambika.swamy@gmail.com and institution mail id: nagalambika_cs@msrcasc.edu.in

Dr. **L. Manjunatha Rao** holds an MCA, MBA, M.Phil, and Ph.D., and serves as a Professor at Master of Computer Applications department at Dr. Ambedkar Institute of Technology, Bengaluru. He has awarded Ph.D from Vinayaka Mission University, Tamil Nadu and obtained Ph.D degree from SV University, Tirupati, Andrapradesh. Dr. Rao's research interests include software engineering and e-Governance, with 36 publications and 61 citations to his credit. He can be contacted at email: manjuarjun2004@yahoo.com and institution mail id: manju.mca@drait.edu.in