Impact of Team Structure on Software Development Productivity and Quality in Pakistani Software Houses

Sadia Khalid[1], Ali Afzal Malik[2]
[1]University of Management and Technology, Lahore, Pakistan; sadia.khalid@umt.edu.pk
[2]National University of Computer and Emerging Sciences, Lahore, Pakistan; ali.afzal@nu.edu.pk

***Abstract*—We are living in the age of information technology and software has become the driving force behind almost everything being done all around the world. The people developing this software are, thus, the real assets in the present world; ordinary people whose brains are working day and night to bring the extraordinary technical touches in our day-to-day lives. Therefore, taking care of these valuable assets and organizing them in the right way is the key to success. Team structure is a critical aspect of the software-development process. Software houses all around the world follow certain team structures to make the best use of their employees' working abilities. This research focuses on the team structures prevalent in software houses in Pakistan and their impact on the productivity of employees and quality of software. A number of surveys were conducted to see how teams are organized and managed in software houses in Pakistan and to gather views on what changes in the current structures would yield better productivity and quality. Then, based on the findings of the surveys, experiments were conducted to see how much the factors pointed out really affected the productivity and quality.**

Keywords—Pakistani software industry, software development productivity, software quality, team structure.

I. INTRODUCTION

The basic goal of every software development company is to develop efficient, reliable, and accurate software products within the allocated budget and time so that the customers can be satisfied. The achievement of this goal, among other things, depends on appropriate distribution of responsibilities, grouping of team members, and communication channels. These factors play a key role in deciding the fate of the product under development.

Software development is a people-intensive exercise. Therefore, the ultimate asset of a software house is the employees working there: business analysts, system architects, programmers, quality assurance engineers, etc. Investing in the right people and getting the best possible return out of them is what an organization has to make sure [1]. This is where the crucial part of management begins. Keeping in view the talent, expertise, and capacity of each individual, a perfect setting needs to be provided to each employee in order to get the maximum out of him/her. This includes forming the right teams, defining appropriate roles, training each employee, and then making sure that the work proceeds in the correct manner.

As simple as it may sound, formation of a team where everyone has their own well-defined roles and responsibilities, is a very difficult task. A slight misjudgment can act as a catalyst to project failure. Every member of a team should have his own priorities and goals, but altogether, each individual's goals must contribute towards the organizational goals. The contribution of each member, however, might be affected by factors both internal (e.g. difficulty of the task assigned, interest in the work, etc.) and external (e.g. peers, management, office environment, etc.) [2].

Like the rest of the world, Pakistan's IT industry is enjoying a bloom. Many software houses of different scales are working all over the country. They deal in clients from both within and out of the country. Being a part of this globalized industry, these software houses try to adhere to international guidelines, standards, and procedures for software development. Similarly,

certain team structures need to be adopted for a more managed approach.

This research focuses on the team structures being used in software houses in Pakistan. It looks at the communication patterns, the responsibility (or work) allotment, the decision making powers, etc. Also, the impact of these structures on software development productivity and quality is explored and verified in this research.

The rest of the paper is organized as follows: Section II briefly describes the basic concepts used in this research. Section III describes the related work done previously on team structure and its relation with productivity, quality and other factors. Section IV presents the problem statement and section V deals with the surveys conducted for this research and the analyses of their results. Section VI describes the experiments conducted to verify the findings of the surveys. Section VII sheds light on the major threats to the validity of this research. Finally, section VIII summarizes the major conclusions while section IX presents directions for future work in this field.

II. BASIC CONCEPTS

A. Software Development Productivity

In general terms, productivity is the relation between amount of goods and/or services produced and the labor and/or expense that goes into producing them [3]. So, generally, software productivity is the ratio between the amount of software produced to the labor and expense of producing it [3].

Software productivity is maximized by developing the most suitable software in the most suitable manner. In other words, giving the customer what he demanded by utilizing minimum resources and time maximizes productivity. If the working hours of a developer justify the functionality implemented by him, the productivity is good. Any imbalance in the two can cause an increase or decrease in the productivity [4].

Therefore, hiring the right people or placing the hired people in the right places is very important for improving the productivity. Other ways to improve productivity include encouraging reuse and having a more defined workflow [5].

B. Software Quality

Software quality can be defined as the degree to which the developed software meets the users' needs or expectations [6]. However, in practical settings, these needs and expectations are not always clear and complete. Furthermore, along with conformance to documented services, quality of software is based on a number of other factors, such as adaptability, portability, reliability, complexity, efficiency, safety, etc. [1].

C. Team Structure

A team structure tells the pattern of communication to be followed and also the people responsible for decision making [2]. Sometimes, clearly defined roles and responsibilities are allocated to team members to avoid any confusions and mishaps during the developmental process [2]. Generally, the structure selected for a team depends on a number of factors such as size and complexity of the project.

A number of team structure classifications have been presented and tested over time by development in various software houses all over the world. One such classification, given by Constantine, is based on four paradigms [7]:

1. Closed paradigm: traditional hierarchy; less innovation
2. Random paradigm: loosely bonded teams; best for following unordered process
3. Open paradigm: consensus based decision; heavy communication; innovative
4. Synchronous paradigm: little communication; teams formed according to work division

One of the oldest team structures is Constantine's closed paradigm [7]. Another team structure, called chief programmer team, was first proposed by Harlan D. Mills around 1970 and was implemented by F. T. Baker [2, 7]. It has a chief programmer that takes care of all the technical details; an assistant that looks after the chief programmer's work; a librarian that looks after the non-technical details. It is a mixture of two other types of team structures, hierarchical team and egoless team [8].

The egoless team structure is similar to Constantine's open paradigm. It gives decision

making power to all the members of the team. It proposes open and direct communication to reach consensus. Such teams tend to have a more innovative approach towards problem solving but they can run into trouble if complex technical challenges arrive. The relationship of team members is on a more personal scale and thus, personal conflicts can also influence work [8].

The hierarchical team's structure is similar to Constantine's closed paradigm. It proposes a rather strict structure where everyone has to be answerable to a higher authority. This type of team is best suited for projects where everything is very well organized and strict orders have to be followed. Every person needs to take responsibility of his own work and hence, there is very little space for mistakes [8].

In 1981, Mantei reported that Weinberg's egoless team and Baker's chief programmer team were the two most common forms of team organization [9]. The egoless teams were as big as 10 members and were frequently exchanging codes and leadership. They worked best for difficult problems without time constraints. The chief programmer team, on the other hand, had an authoritative hierarchy to follow for decision making as well as communication. It worked best with tight schedules of well-tailored small tasks. The paper looked at another team organization, controlled decentralized, that lied in between the other two. It had hierarchy, it allowed peer communication and it worked best for large, well-tailored, short-timed projects.

III. RELATED WORK

Rising and Janoff studied the utility of Scrum for small teams [10]. They described how they experimented with Scrum on three of their organization's teams. The major conclusion of their research was that work divided into manageable pieces and reviews or meetings conducted in a regular pattern made it easier for small teams to work better.

Pinkowska studied the connection between team cohesiveness and productivity [11]. This study included two cohesion types; social, where non-working relationships were involved and task-based, where bonding was just due to similar work allotment. The research showed team cohesion to be good in certain aspects, such as information flow, behavior control, conformity, etc. but it had its downside as well, including wasting time in socializing, resistance to change, conflicts between team and organizational goals. Some factors impacting cohesiveness were looked upon (e.g. size, success, interaction, etc.), each having its own influence and consequence. The conclusion derived was that high cohesion did have a positive impact on productivity, but at certain stages, it can be bad as well.

In [4], Sudhakar and colleagues discussed why was it important to measure a development team's productivity. The factors that had an impact on productivity were listed, e.g. team communication, average team size, application experience, etc. It showed that improved productivity was to be achieved by improving productivity of individuals and then, of the teams.

Abbas, Gravell, and Wills published agile projects governance survey, conducted in 2009 [12]. The main aim was to determine how the methods and principles, based on the Agile Manifesto [13], helped improve productivity. The agile part of this paper was not directly related to team organization's impact on productivity and quality, yet there were some results based on team size which linked the two. The results of this paper showed that:

- Small-sized teams had more success rate – 72%
- If all team members participated in review meetings after each iteration, the success rate increased – 66%
- When every team member got a fair chance of speaking up his mind in meetings, the success rate increased – 88%

Basri and O'Connor studied some team factors to check their impact on knowledge management process in small companies having small team sizes [14]. The results proved that small teams felt more comfortable working in a flat team structure manner, with direct communication channels and informal management arrangement. These small teams had very high levels of knowledge sharing and the whole setup helped them in their work by preventing misunderstandings and work delays.

In 2011, Cataldo and Ehrlich reported the impact of team communication in geographically distributed teams on productivity and quality [15]. The agile development style was taken into

consideration to base performance measure on work done and quality measure on defects per iteration. Hierarchical team structure was proven to be better for productivity; as few trained people (managers) were handling the work breakdown and distribution responsibility. On the other hand, small teams with flat structure were proven to be more fruitful with quality; as work in progress needed open communication and knowledge exchange between all members of the team.

In short, multiple team structures have been proposed over time; each having its own homeground. Similarly, the factors affecting the productivity and quality of software have been looked upon from various angles. Attributes have been defined and different techniques have been formulated to measure the productivity and quality of software.

Like the rest of the world, Pakistan is contributing its share of software and system development to the IT sector. However, there has not been much research on the way software teams are structured in Pakistani software houses and the impact of these structures on productivity and software quality.

IV. PROBLEM STATEMENT

This research is focused on answering the following questions:

1. What is the most commonly used team structure followed in software houses in Pakistan?
2. Which team structures are endorsed by software developers and professionals to maximize productivity and quality?
3. Are the claims made by professionals about team structures supported by empirical evidence?

V. SURVEYS AND ANALYSES

In order to find the answers to the above questions, three surveys were conducted via online surveying technique; a pilot study followed by a follow-up survey and then based on the findings of the two, a final survey.

Online surveying technique was used because it is an easier way to reach out to maximum respondents, it saves time and most techniques provide you with automated result compilations.

A. Pilot Study

Before launching the full-fledged survey, a questionnaire was formulated to find the most common team structures and their associated characteristics in software houses in Pakistan. The questionnaire was given to 11 people, working in 9 different software houses in Lahore (Pakistan). All the participants had a minimum of 16 years of education and, at least, 3 years of experience in the software industry. The results of this pilot study showed that:

- Team size of up to 15 members was preferred, irrespective of the organization being big or small; having, at max, 3 domain experts and 3 new recruits
- Meetings were conducted after almost every major software development life cycle stage
- Open door policy (egoless structure) was mostly followed
- It was safe to express contradictory opinions openly and without making enemies amongst the team members
- Work decisions were mostly project dependent
- The roles of team members were fixed but the project manager changed with almost every project

Based on the results of the pilot study and literature review, the following characteristics defining team structure were identified:

- Leadership and management roles
- Hierarchy of the team
- Interaction and communication patterns
- Decision making authority
- Domain knowledge of the team members
- Size of the team

The following project and people factors that affect team structure were also identified:

- Roles assigned to the team members
- Size of the team
- Overall productivity of the project
- Overall quality of the project

B. Follow-Up Survey

In order to check the relevance of the factors that were being considered, a follow-up survey was designed and dispatched to 8 people, working in 8 different Pakistani software houses. All the participants had a minimum of 18 years of education and were in the software industry for

more than 3 years. The summary of the responses is given in Table I.

TABLE I. SUMMARIZED RESPONSES OF FOLLOW-UP SURVEY

	Factors identified in both pilot study and follow-up survey	**Additional factors identified in the follow-up survey**

Characteristics that define team structure	• Roles
	  ○ Leadership
	  ○ Management
	• Hierarchy
	• Interaction & Communication
	• Decision making
	• Domain Knowledge
	• Size
		• Capabilities
		  ○ Technical Skills
		  ○ Experience
		• Dedication
		• Objectives
		• Division of work

PROJECT factors affected by a team structure	• Productivity
	• Quality
		• Work Load
		• Price
		• Timeframe
		  ○ Management
		  ○ Delivery of releases
		• Error rate
		• Cohesiveness
		• Reviews and Inspection Meetings

PEOPLE factors affected by a team structure	• Roles • Size	• Homogeneity • Work Load • Behavior • Groupthink

As shown by the responses in Table I, the factors which were identified earlier in the pilot study for defining team structure and their impact on the software development productivity and quality were mentioned by the respondents as well. However, a few more factors (such as experience and error rate) mentioned in the follow-up survey were worth considering.

C. Final Survey

Based on the findings of the pilot study and the follow-up survey, five major characteristics of a team structure (team size, number of domain experts, roles of team members, team decisions and communication initiation pattern) were finalized. A final survey was designed for finding out the existing state of the practice with respect to these five factors and how this practice could be improved to maximize productivity and quality. The response rate of the final survey was 80.8%, out of which 5.8% of the respondents were freelancers and the rest worked in various software houses in Pakistan (see Fig. 1 for city weightage of respondents and Fig 2. for designation of respondents).

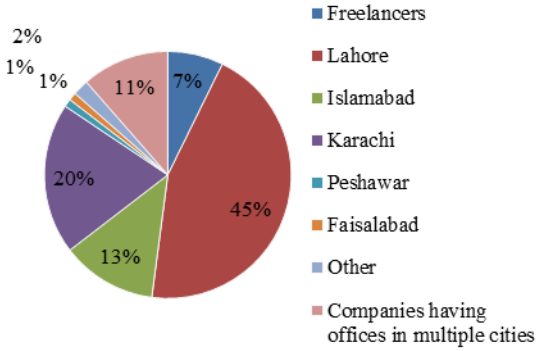


Fig. 1: City distribution of respondents

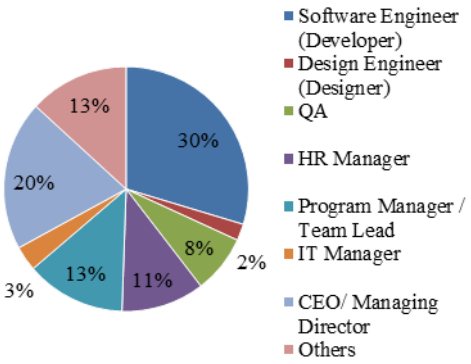


Fig. 2: Designation of respondents

Each practice question was divided into three categories for the respondents to consider, based on their current working environment and their experience:

1. Current Practice – these questions were to be answered keeping in mind the practice being followed in the respondent's current team setup
2. Maximum Productivity – these questions were to be answered keeping in mind the practice which, according to the respondent, will give maximum productivity
3. Maximum Quality – these questions were to be answered keeping in mind the practice which, according to the respondent, will give maximum quality

The survey results showed that the team size deployed majorly in software houses was between 6 and 10. Also, the ideal team size to gain maximum productivity and quality was opted to be the same as the current trend. The

option least answered by the respondents for achieving productivity was 11 to 15, which was meeting on the boundary with the least answered option for achieving quality value, i.e. more than 15.

The number of domain experts in a team was 2 in most of the companies; which was a number less than what the respondents thought to be good for achieving maximum productivity and quality. The least answered option was 5 for the number of domain experts. This result was in collaboration with the desired team size; 3 domain experts for a team of 6 to 10 were acceptable, but not 5 or more.

The roles were said to be project dependent in most of the companies, i.e. they were fixed or rotated among team members based on the project at hand. This approach was also thought to be good for achieving maximum productivity and quality by most of the professionals. However, continuously changing the roles in every project was not considered to be appropriate for good productivity and quality.

Most of the respondents stated that currently, the decisions were being communicated by the team lead in their companies. However, they believed that consensus was a better option to achieve maximum productivity and quality. The results also showed that currently, senior to junior communication was the most common trend in the companies; since decisions were being communicated by the team lead. Peer to peer communication was preferred for better productivity and quality, as the respondents favored consensus based decisions. The results of this final survey are summarized in Table II.

It shows that the options which were opted by most of the respondents were considered to be the best practices for achieving maximum productivity and maximum quality. The options which were least opted by the respondents were thought to be not-good for better productivity and quality and were placed in "Minimum Productivity" and "Minimum Quality" category.

VI. EXPERIMENTS AND RESULTS

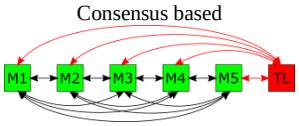
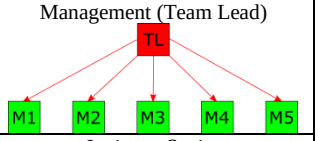
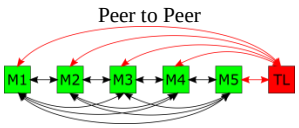
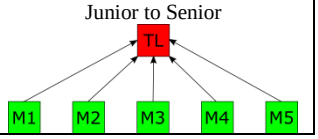
Based on the findings of the final survey, a 7-day experiment was designed to verify the results. The maximum productivity and maximum quality values (see Table II) were combined together to form Team A, and the minimum productivity and minimum quality values (see Table II) were combined to form Team B.

Team setup for both the teams is summarized in Table III. Since the best value received for team size was 6 to 10, including team lead, a team size of 6 was decided for both the teams. The team lead was the domain expert of the team. One domain expert, instead of 3, was selected because for a team of 6, a single domain expert looked sufficient. The roles in both the teams were decided to be fixed for a 7-day experiment. Only the team lead was allowed to play multiple roles of the domain expert as well as the team lead. The designing decisions and the technical and non-technical decisions were to be made by consensus in Team A. For Team B, they were to be made by the team lead and be communicated to the rest of the members. In Team A, the members were allowed to talk and discuss among themselves, formally as well as casually. But in case of Team B, members were forbidden from engaging in all sorts of sideways communication. In case of any help, they were to consult the team lead only.

TABLE II. SUMMARY OF FINAL SURVEY

	Current State of Practice	**Maximum Productivity**	**Minimum Productivity**	**Maximum Quality**	**Minimum Quality**
Team Size	6 – 10	6 - 10	11 – 15	6 - 10	15+
Domain Experts	2	3	5	3	5
Team Decisions	By team lead	Consensus of team members	By higher management	Consensus of team members	By higher management
Definition of Roles	Project dependent	Project dependent	Any member can get any role at any time	Project dependent	Any member can get any role at any time
Communication Initiation Pattern	Senior to Junior	Peer to Peer	Junior to Senior	Peer to Peer	Junior to Senior

TABLE III. SETUP FOR TEAM A AND TEAM B

		Team A	Team B
Constant Values	Team Size	6	6
	No. of Domain Experts	1	1
	Roles	Fixed	Fixed
Variable Values	Decisions	Consensus based 	Communicated by Higher Management (Team Lead) 
	Communication Initiation Pattern	Peer to Peer 	Junior to Senior 

The experiment was run in two different companies. Company ABC was a 7-year old web application development company also providing IT solutions to worldwide clients. They had a total of 20+ employees, with an average experience of 5 years. Company XYZ was a 5-year old web application development company dealing in website designing and development and UI and print media designing. They had a total of 20+ employees, with an average experience of 4 years. The purpose of the experiment was to verify the views of the professionals.

Both the teams were trained before the experiment about their setup. The data to be analyzed from the experiment was to be gathered according to Table IV.

TABLE IV. ANALYSIS TABLE FOR THE EXPERIMENT

	Team A	Team B
Time log profile		
1. Graphic Designing		
2. HTML Conversion		
3. Development		
Lines of code		
Number of defects identified in the final product (via Black Box Testing)		
Number of features and sub-features missing from the final product		
Communication problems encountered (if any)		

The teams were to be evaluated on the basis of the following metrics:

Avg. Team Productivity = No. of Features/Sub-features Developed Successfully

Avg. Programmer Productivity = Lines of Code/ Total Time taken by Programmers

Total Defect Density = No. of Defects/ KLOC

The defects were also to be categorized according to four severity levels [16]:

1. Critical: defects that were impairing one or more critical functionalities of the project and were without a workaround
2. High: defects that were impairing one or more fundamental functionalities of the project but with a workaround
3. Medium: defects that were impairing one or more minor functionalities of the project but with a workaround
4. Low: defects that were not affecting any functionalities of the project but were only cosmetic

The defect density of each severity level was also to be calculated using the following metrics:

Critical Defect Density = No. of Critical Defects/ KLOC

High Defect Density = No. of High Defects/ KLOC

Medium Defect Density = No. of Medium Defects/ KLOC

Low Defect Density = No. of Low Defects/ KLOC

Since, Team A had all the points that were thought to lead to better software development productivity and quality, it was expected that Team A will have better productivity and quality vis-à-vis Team B.

Both the teams were given an e-commerce website each, which was to be developed within the given 7 days.

A. Company ABC

According to the results of the experiment, Team B, unexpectedly, had a better team setup and was able to complete more features/sub-features with lower number of defects. Out of the 8 required features, Team A was able to complete 4 and Team B was able to complete 7 features successfully. Hence, Team B was approximately 1.8 times more productive than Team A. Similarly, the programmer productivity of Team A was 43.4 and of Team B was 52.2; Team B had approximately 1.2 times more productive programmers than Team A. The total defect density of Team A was 4.1 and that of Team B was 0.4; Team A had approximately 10.3 times more defects than Team B. Fig. 3 shows the comparison of both the teams with respect to team productivity while Fig. 4 shows the productivity comparison of both the teams with respect to programmer productivity. Fig. 5 shows comparison of both the teams with respect to quality.

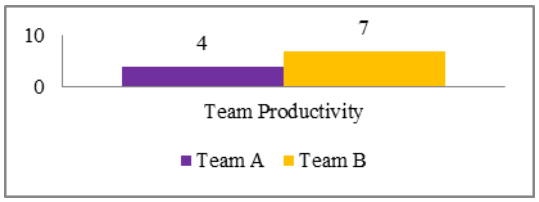


Fig. 3: Team Productivity in Experiment 1 of Company ABC

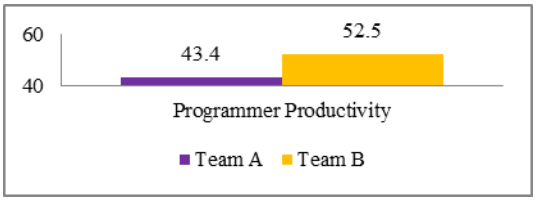


Fig. 4: Programmer Productivity in Experiment 1 of Company ABC

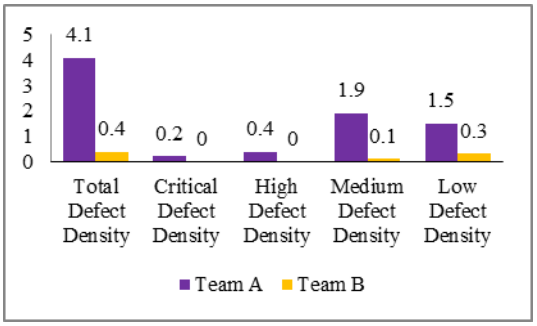


Fig. 5: Defect Density and Severity in Experiment 1 of Company ABC

The results from this experiment were not in conformance with the survey results and the earlier expectations. The reason behind this was that the setup being traditionally followed in Company ABC was that of Team B. The subjects were used to taking orders from higher management and applying them directly.

An important question that arose here was that if Team A had a familiar setup, what would have been the results? In order to see these results, a second experiment was conducted in Company ABC with the same team setups but a different e-commerce project. Since Team A had now worked with the setup for 7 days, they were now familiar with it.

The second experiment fulfilled the expectations and Team A showed better results as compared to Team B. The results of the second experiment are shown in Fig. 6, Fig. 7 and Fig. 8. Out of the 16 required features, Team A was able to complete 14 and Team B was able to complete 11 features successfully. Hence, Team A was approximately 1.3 times more productive than Team B. Similarly, the programmer productivity of Team A was 24.8 and of Team B was 23.2; Team A had approximately 1.1 times more productive programmers than Team B. The total defect density of Team A was 1.1 and that of Team B was 2.5; Team B had approximately 2.3 times more defects than Team A.

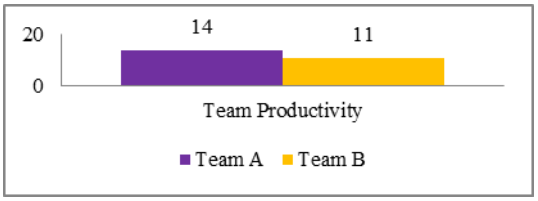


Fig. 6: Team Productivity in Experiment 2 of Company ABC

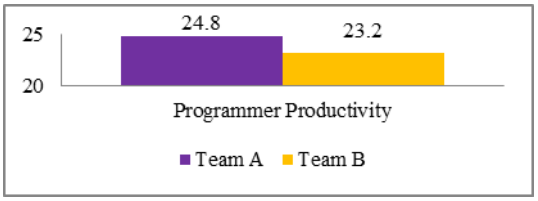


Fig. 7: Programmer Productivity in Experiment 2 of Company ABC

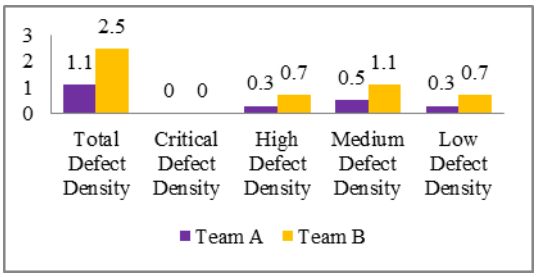


Fig. 8: Defect Density and Severity in Experiment 2 of Company ABC

B. *Company XYZ*

According to the results of the experiment in company XYZ, Team A had a better team setup and was able to complete more features/sub-features with lower number of defects. Out of the 16 required features, Team A was able to complete 6 and Team B was able to complete 2 features successfully. Hence, Team A was approximately 3 times more productive than Team B. Similarly, the programmer productivity of Team A was 48.2 and of Team B was 32.9; Team A had approximately 1.5 times more productive programmers than Team B. The total defect density of Team A was 1.2 and that of Team B was 6.1; Team B had approximately 5.1 times more defects than Team A. Fig. 9 shows the comparison of both the teams with respect to team productivity while Fig. 10 shows the productivity comparison of both the teams with respect to programmer productivity. Fig. 11 shows comparison of both the teams with respect to quality.

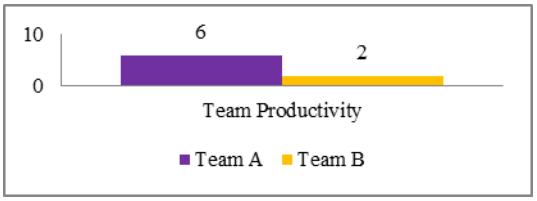


Fig. 9: Team Productivity in Experiment of Company XYZ

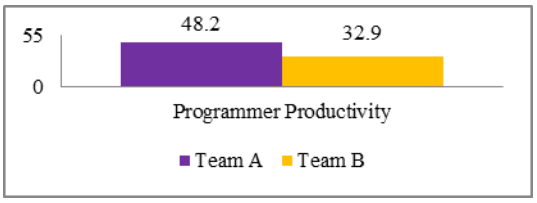


Fig. 10: Programmer Productivity in Experiment of Company XYZ

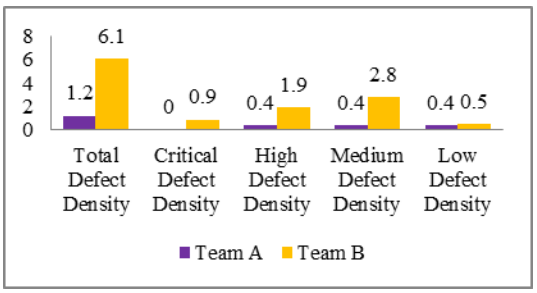


Fig. 11: Defect Density and Severity in Experiment of Company XYZ

The results from this experiment were in conformance with the survey results and earlier expectations because the setup being traditionally followed in Company XYZ was that of Team A and the subjects were used to consensus based decision making. However, unlike Company ABC, Company XYZ did not permit for a second experiment to check if the results would be different if Team B's setup was somewhat familiar to the team.

VII. THREATS TO VALIDITY

Though the findings of the survey were tested and verified with experimentation, yet there are some threats to validity. The factors considered for the final survey i.e. team size, number of domain experts, team decision making powers, definition of roles and communication initiation pattern, were selected after research findings; there may be other factors, e.g. work experience and organization size, which are directly or indirectly related to team structure and its impact on software development productivity and quality.

Also, of the five factors from the final survey, 3 were made constant for the experiment i.e. team size, number of domain experts and roles; if

made variable, each of them might have a certain impact on the final results.

The experiment results for Company ABC changed once Team A became familiar with their new team setup. The same could not be tested for Company XYZ. Hence, there is a chance of Team B's performance to be improved after their familiarity with their team setup increases.

VIII. CONCLUSIONS

Formation of teams is no doubt a very crucial and important part of any software house. Apart from the working environment and individual intellect of the team members, the structure of the team counts a lot in the overall success of projects being carried out. This setup can indeed vary from organization to organization and from project to project, but some basic rules and facts are observed and experienced by people working in this domain for some time now.

This study gathered views from professionals working in software houses in Pakistan about team structures and factors influencing productivity and quality. For maximum productivity and quality, the majority voted for a team size of 6 to 10, with 3 domain experts. The team members having the decision making power was preferred over the orders coming from a higher authority. Project dependent roles were favored over completely fixed or variable roles.

The results were tested and verified by conducting experiments in two different companies. The experiments provided evidence indicating that the factors opted by professionals were indeed capable of improving productivity and quality. Hence, if these outcomes are kept in mind while defining a team structure, the results can be profitable and rewarding.

IX. FUTURE WORK

The work of this research can be extended by conducting further experiments following the team structures defined in the research. Also, the factors that were kept constant for the teams such as team size, number of domain experts, and roles of team members can be made variable to observe the impact of each factor individually.

REFERENCES

[1] I. Sommerville. "Managing People", in Software Engineering, 5th ed., Ed. USA: Addison-Wesley, 1995, pp. 567-576.

[2] H. V. Vliet. "People Management and Team Organization", in Software Engineering Principles and Practice, 3rd ed., Ed. NJ: John Wiley & Sons, May 2008, pp. 97-112.

[3] S. A. Dilawer. "Productivity", in Practical Guide of Software Development Project Management in Practice, 1st ed., Ed. USA: lulu.com, December 2011, pp. 152.

[4] G. P. Sudhakar, A. Farooq and S. Patnaik. (2011, Sep.). Measuring Productivity of Software Development Teams. Serbian Journal of Management *7(1)*,pp. 65- 75.

[5] R. W. Selby. "Software Economics", in Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research, Ed. NJ: John Wiley & Sons, June 2007, pp. 161-171.

[6] J. Gao, H.-S. J. Tsao and Y. Wu. "Quality Assurance for Software Components", in Testing and Quality Assurance for Component-based Software, Ed. USA: Artech House, January 2003, pp. 293.

[7] R. S. Pressman. "Project Management", in Software Engineering, A Practitioner's Approach, 6th ed., Ed. NY: McGraw-Hill Publishing Company, 2005, pp. 629-630.

[8] T. C. Lethbridge and R. Laganière. "Managing the Software Process", in Object-Oriented Software Engineering: Practical Software Development using UML and Java, 2nd ed., Ed. NY: McGraw-Hill Publishing Company, 2001, pp. 445-449.

[9] M. Mantei. (1981, Mar.). The Effect of Programming Team Structures on Programming Tasks. Communications of the ACM. *24(3)*, pp. 106-113.

[10] L. Rising and N. S. Janoff. (2000, Jul-Aug.). The Scrum Software Development Process for Small lTeams. IEEE Software. *17(4)*, pp. 26-32.

[11] M. Pinkowska. "IT Software Project Management: Impact of Team Cohesiveness on Productivity and Performance", in Proceedings of 14th IDIMT, September 2006, pp. 75-90.

[12] N. Abbas, A. M. Gravell and G. B. Wills. "The Impact of Organization, Project and Governance Variables on Software Quality and Project Success", in Agile Conference (AGILE) , August 2010, pp. 77 - 86.

[13] http://agilemanifesto.org/

[14] S. Basri and R. V. O'Connor. (2011, Jul.). The Impact of Software Development Team Dynamics on the Knowledge Management Process. 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE 2011). Available: http://doras.dcu.ie/18654/1/Basri-OConnor-SEKE20011.pdf

[15] M. Cataldo and K. Ehrlich. "The Impact of the Structure of Communication Patterns in Global Software Development: An Empirical Analysis of a Project Using Agile Methods", Institute for Software Research, Carnegie Mellon University, May 2011.

[16] S. Naik and P. Tripathy. "System Test Execution", in Software Testing and Quality Assurance: Theory and Practice, Ed. Canada: John Wiley & Sons, 2011, pp. 411.

ABOUT THE AUTHORS

Sadia Khalid started her professional career in 2011 as a Search Engine Optimization (SEO) engineer. She was a Fulbright scholar in her BS and graduated with Magna Cum Laude (honors) with double majors in software engineering and information technology from Forman Christian College University in 2011. Her research work in her M.S. focused on areas related to the impact of team structures on software productivity and quality in the software industry of Pakistan. Her first teaching appointment was in the year 2014 at Forman Christian College University as a visiting faculty member. Later on, she held the position of a full-time faculty member at the University of Management and Technology (UMT), Lahore, Pakistan. Her current interests include software project management and software ethics.

Dr. Ali Afzal Malik started his professional career in 2003 working as a software engineer in Techlogix – a well-reputed Pakistani software house. After receiving the prestigious Fulbright scholarship in 2005, he obtained M.S. and Ph.D. degrees in Computer Science from the University of Southern California (USC), Los Angeles, USA, in 2007 and 2010, respectively. He was awarded the Office of International Services (OIS) Academic Achievement Award twice (2007 & 2010) during his stay at USC. His research paper on the quantitative aspects of requirements elaboration was given the best paper award in SBES 2008 (Sao Paulo, Brazil). Before joining the National University of Computer and Emerging Sciences in 2013, Dr. Malik has held an adjunct faculty position at the Lahore University of Management Sciences (LUMS) and a full-time faculty position at the University of Central Punjab (UCP). He has undertaken research in software cost estimation at two of the world's leading research centers in software engineering i.e. USC's Center for Systems and Software Engineering (CSSE) and Institute of Software, Chinese Academy of Sciences (ISCAS). His current research work focuses on empirical software engineering, software cost estimation, and software process improvement.