Journal of Information Systems Engineering and Management
2020, 5(3), em0117
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

MODESTUM
OPEN ACCESS

An Empirical Study to Investigate the Effect of Transactive Memory System on Software Development Team Performance

Hamsheed Salamut [1], M Yasser Chuttur [1*]

[1] University of Mauritius, Reduit, Moka, MAURITIUS
***Corresponding Author:** y.chuttur@uom.ac.mu

Citation: Salamut, H. and Chuttur, M. Y. (2020). An Empirical Study to Investigate the Effect of Transactive Memory System on Software Development Team Performance. *Journal of Information Systems Engineering and Management, 5*(3), em0117. https://doi.org/10.29333/jisem/8372

ARTICLE INFO

Published: 30 Jun. 2020

ABSTRACT

Software development often involves teams of developers interacting with each other to develop a software product or service. Given that individuals in the same team may have varying expertise and knowledge, it is believed that proper management of Transactive Memory System (TMS) or group knowledge within a team is more likely to ensure optimum team performance. In this study, we test this hypothesis by using an experiment in which we compare the performance of different teams with different TMS in solving the same software related problems. Our results indicate it is under certain conditions, that teams with higher TMS will usually achieve better performance than teams with lower TMS.

Keywords: transactive memory systems, software development team, experimentation

INTRODUCTION

Teams are principal instruments to leverage diverse expertise from individual team members towards a common goal (Akgün, 2020; Lewis, 2003). Teams may consist of a small or large number of individuals, all carrying "complementary skills and who are committed to a common purpose, set of performance goals and approach for which they consider themselves mutually accountable" (Ryan and Connor, 2019; Smith-Jentsch et al., 2001).

In software development, it is therefore common to find teams working on software projects, which consists of tasks that cannot be completed by one individual alone. In an attempt to ensure project success, a software development team is set up such that different levels and distribution of expertise exist within the team (Ryan et al., 2019; Walz et al., 2003). However, as argued by Faraj and Sproull (2000) and Rapp et al. (2019), the mere presence of individuals with varied expertise is not sufficient for a team to achieve optimal performance. Additional consideration must be given to the sharing of knowledge among members. In other words, all members of a team must be well acquainted with the expertise of each other so as to be able to capitalize on each individual expertise when solving the same problem.

Transactive Memory Systems (TMS) is a mechanism through which members of the same team maintain a common cognitive directory of the abilities and expertise possessed by each team member (Wegner, 1986). TMS is part of a larger body of research concerning team cognition whose main focus is on knowledge sharing and team performance (Smith-Jentsch et al., 2001). Conceptually, team cognition can be regarded as a set of mental models collectively possessed by a group of individuals that allow them to complete tasks by acting as a coordinated unit (He et al., 2007). In order to enhance overall team performance, team cognition encourages optimum teamwork by adjusting member behaviors and actions in a collaborative fashion.

Given the high level of collaboration required in most software projects, team cognition within teams composed of software developers cannot therefore be overlooked. However, we note that despite the importance of team cognition, research on TMS is scarce in the field of software development such that little is known on the relationship between TMS and software development team performance. In this study, we aim at bridging this gap by providing some details on TMS and evaluating its effect on the performance of software developers in solving the same problem. We also consider problems complexity so as to have a broader understanding of the effect of TMS on team performance when faced with problems of varying complexity.

SOFTWARE DEVELOPMENT

Software development is a process, which results in the production, deployment and/or maintenance of a software product (Ostaysi et al., 2019). Despite the fact that the process in itself is well defined as different stages such as planning, requirements gathering, design, implementation, testing and deployment (Sommerville, 2019), the success of a software project is not always guaranteed. Several factors such as requirements clarity; time frame for delivery; available budget, expertise, etc. are well known to influence the outcome of a software project (Dhir et al., 2019; O'Connor, 2008; Schnabel et al., 2006).

In this study, the focus will be on human factor and in particular on teams. As argued by O'Connor (2008), software development not only involves effective programming and tools, but it also depends highly on individuals. Similarly Beaver and Schiavone (2006) claim that people are the principal drivers to achieve software quality in a software project. Other studies have also demonstrated the importance of considering human related factors in software development. For an extensive literature review, see (Laughery et al., 1985). Despite an abundance of literature on human factors and software development however, it is noted that little is known on the relationship between team performance and team cognition warranting further research in that area.

Software Development Teams

A team consists of a small number of individuals with "complementary skills who are committed to a common purpose, set of performance goals and approach for which they consider themselves mutually accountable" (Katzenbach et al., 1993). Teamwork implies that individuals work in a collaborative environment to achieve a common goal by developing and sharing knowledge, skills and being adaptable to undertake various roles (Terricone et al., 2002).

Software development teams differ from other project teams (Trendowicz et al., 2008). In a software development team, the focus is mainly about team performance and team productivity. As stated by Gorla and Lam (2004), a well-composed team is vital to achieve better team performance, which in turn is determinant for software project success. Given that the nature of software development projects constantly evolves due to globalization and dispersion across several sites, creating well-composed teams becomes a necessity and project managers have to regularly adopt various strategies when building the right team for the right project (Ebert et al., 2001).

Setting up Software Development Teams

Ahn et al. (2007) introduced a method relying on three dimensions to compose a software team. The dimensions are: Reliability: the extent to which a team member is able to fulfill the commitment; Quality: the quality of product a team member is able to deliver; and Availability: the degree to which a team member is able to work collectively with other individuals. These three dimensions are evaluated for each potential team member. A weight is then applied on these dimensions based on the importance of these dimensions for the actual project. The relevance for each potential team member is then calculated as the weighted sum of these three dimensions. The team members with the highest scores are then chosen to work on the project. Following a study on how project managers set up software teams in the software industry, França et al. (2009) later listed the following set of criteria considered important when selecting team members.

- **Technical profile,** which reveals the individual's technical knowledge in a specific technology or tool. This criterion also encompasses the knowledge associated to business process areas of software. The technical profile indicates if the individual has the ability to work in a project, without taking into consideration his experience and productivity.
- **Individual costs,** which indicates the costs of each individual recruited by the organization, and the impacts of each professional in the project budget.
- **Experience and Productivity,** which focuses on the professional experience of each individual based past productivity rates. It provides an indication of what the individual is able to deliver in a given amount of time.
- **Personality,** which refers to the individual differences in characteristic patterns of thoughts, characters and feelings by each individual that influences the individual's behaviors and motivation in various circumstances.
- **Availability,** which relates to the amount of time that the individual can work on the project.
- **Behavior,** which indicates the set of actions and reactions observed by the individual in certain situations and in relation to the environment.
- **Project importance,** which focuses on the strategic, competitive and financial importance the project has to the organization at the time the team is being formed;

Da Silva et al. (2013) further conducted a systematic analysis of the team building criteria by looking into specific literature on organizational psychology and the authors categorized the criteria identified from literature into two main factors namely individual factors and organizational factors. Da Silva, et al. argued that some individual factors such as technical characteristics can be modified through training but innate characteristics, which are fairly stable, remain difficult to change. Da Silva et al. also indicated that in order to develop better conditions for software project success, organizations must embrace more comprehensive set of teams building criteria, and make them explicit and more recognized within the organization.

In general, it is clear that several strategies exist to set up team members. However, most, if not, all criteria used to select team members focus exclusively on individual abilities with little consideration for team cognition and whether individual members will be able to work collectively towards optimum team performance.

Software Development Team Performance

Team performance can be defined as the extent to which a team is able to achieve established cost, time and quality goals (Hoegl et al., 2001; Salas et al., 2008). According to Ong and Kankanhalli (2005), team performance encompasses both subjective and objective measures. In the context of software development, examples of objective measures include adherence to schedules and costs whereas examples of subjective measures include ratings of team performance by teammates such as user satisfaction and teamwork satisfaction.

It follows that several factors are known to affect team performance. As detailed by Sudhakar, et al., (2011), factors influencing team performance fall into four categories, namely: 1) Technical; 2) Non-Technical; 3) Organizational; and 4) Environmental factors.

Technical factors

Ong and Kankanhalli (2005) pointed out that project size, project complexity, team composition, team processes and collective expertise are the main factors affecting a software development team's performance. In addition, inconsistent software development environments such as source codes, build tools, debugging tools have been regarded as major issues for large software development projects that affect team performance (2011). Blasi et al. (2008) further reported that the knowledge, skills and experience of team members also have an impact on team performance.

Non-technical (soft) factors

Sudhakar et al. (2011) expressed that team diversity, team climate and team member's capabilities are all part of non-technical factors that influence a team performance. Acuña et al. (2008) further indicate that team performance can be predicted by the interactions occurring between team members and on the team members' constructive or cognitive representation of their work surroundings.

Organizational factors

Sudhakar et al. (2011) listed organizational climate, organizational culture, organizational structure, and organizational values as major influencers of team performance. The members of the software development team should be familiar with the native and foreign cultures if they are overseas. Hence, if a software development team is diversified culturally, there will be an impact on the team's performance.

Environmental factors

Cohen and Bailey (1997) reported that team performance is governed by external factors such as organizational settings and instabilities. The authors argued that environmental factors, design factors, group personality traits, and internal and external processes all forecast a team's performance. Team members' participation in meetings, inadequate language skills, and various legal restrictions associated to organizational work timings and unanticipated environmental maintenance actions can have an influence on the performance of the team as well (Kozlowski et al., 2000).

A close review of all four categories of factors influencing team performance reveals that knowledge of each team member's abilities and limitations is an important consideration when setting up a team. In this study, we focus on the consideration for transactive memory systems as a measure of team cognition when setting up teams.

TEAM COGNITION AND TRANSACTIVE MEMORY SYSTEMS

Team cognition is conceptualized as the mental models collectively possessed by a group of individuals when completing a given task as a coordinated unit (He et al., 2007). Team cognition is known to encourage communication in teams and behavior adjustments in a collaborative fashion in order to enhance overall team performance. In software development, team cognition is essential in teams. Project managers can make use of team cognition for efficient management of each member's knowledge and expertise and to coordinate activities related to the requirements of a given project (He et al., 2007).

Transactive memory as conceptualized by Wegner (1986), consists of a group of individual memory systems that incorporates both the individual expertise and team expertise required to provide solutions to problems. Transactive memory emphasizes on the expertise possessed by team members such that the expertise of the members within the team can lessen individual cognitive functions. Consequently, members can coordinate and depend on other members' knowledge in an easy and efficient manner.

In transactive memory, the cognitive division of labor in groups is considered to consist of two components namely: 1) internal memory, i.e., what individuals know personally and 2) external memory, i.e., what individuals collaboratively know about the knowledge of other team members. While transactive memory exists in the mind of individuals, a Transactive Memory System or TMS is a collective construct that is present among individuals as a function of their individual transactive memories (Kozlowski et al., 2000). To clarify on the subject, Wegner further, inform us that a TMS consists of three stages as follows.

Encoding

In the encoding stage, the team members obtain information on the other members' domains of expertise and classify it by attributing each knowledge domain to the corresponding team member. Usually, this acquaintance unfolds through "who do what" conversations. The encoding process takes place through interaction between team members such as through knowledge sharing and soliciting information from other teammates.

Table 1. 2X2 Experimental Design

	TMS	
Problem Complexity	**Low**	**High**
Low (P_{Low})	Group A (n = 10)	Group B (n = 10)
High (P_{High})	Group A (n = 10)	Group B (n = 10)

Storage

In the storage stage team members who possess corresponding expertise capture related information. Once the experts have been recognized, new information is conveyed directly to the related team member, a process, which enhances the learning process and lessens the load on the memory of the individual member.

Retrieval

During the retrieval phase, a team member makes use of the developed transactive memory to locate a team member who specializes in the required knowledge field and then revert to that member to acquire the knowledge.

The effect of TMS on team performance has been tested in different settings such as in Akgun et al. (2006), Zhang et al. (2007), Moreland and Myaskovsky (2000), Rulke and Rau (2000), Li et al. (2015), Dai et al. (2017), Tsai et al. (2016), and Wang et al. (2018). All of those studies unanimously reported that teams with a strong TMS, i.e., group knowledge of each individual's knowledge is more likely to lead to better team than teams with poor or low TMS where each individual member is unaware or have little knowledge of other team member's knowledge.

RESEARCH QUESTIONS AND HYPOTHESES

When setting up teams for software development, little consideration is given to TMS. Past studies in areas other than software development demonstrate that TMS is an important factor to consider when setting up team as this can lead to better team performance. In this study we aim at answering the following research question:

Is there a relationship between transactive memory system and team performance when developing software?

Taking into consideration TMS as the independent variable and team performance as the dependent variable, we formulate our null and alternative hypotheses as follows:

HA_0: There is no relationship between TMS and software team performance when developing software.

HA_1: There is a relationship between TMS and software team performance when developing software.

However, we posit that problem complexity can also have an effect on the experimental outcome. This is because different complexity could yield different results as individuals may require different level of interactions. Consequently, we further formulate the following hypotheses:

HB_0: There is no relationship between task complexity and software team performance when developing software.

HB_1: There is a relationship between task complexity and software team performance when developing software.

HC_0: There is no interaction between TMS and task complexity when developing software.

HC_1: There is interaction between TMS and task complexity when developing software.

To answer our research question, we proceed by adopting an experimental design approach where we compare the team performance of two teams, one with low TMS against another one with high TMS in solving the same programming problem. Details on our experiment are given further.

EXPERIMENTAL SET UP

A 2X2 factorial experimental design was adopted for this study. A total of 60 participants were recruited for this experiment. Thus, 20 teams consisting of 3 individuals each could be distributed into two groups (A and B) of 10 teams (n=10) each based on 1) the TMS measured in the team (high/low) and 2) the complexity of the problem to be completed (low/high). **Table 1** shows the distribution of the two groups for the experiments. In the present experiment, Group A consisted of 30 individuals grouped in 10 teams of 3 individuals and which demonstrated a high level of TMS whereas, Group B consisted of 30 individuals grouped in 10 teams of 3 individuals which demonstrated a low level of TMS.

Participant Recruitment

Participants recruited for the study were undergraduate students who had already completed a two-year degree course in computer science program and who have already followed a programming course in the Java language. Given that the programming skills of an individual may have an impact on a software development team performance (Blasi et al., 2008), we decided to recruit participants of similar programming skills only, here Java. Interested participants were screened following data received from an online form that all participants had to fill in prior to taking part of the experiment. Each participant had to

Specialization	My team members and I have specialized knowledge in specific areas
	I have knowledge about a specific area that my team members do not have
	My team members and I are responsible for expertise in different areas
	The specialized knowledge of both my team members and myself is needed to complete a task
	I know whether it is my team members or myself who have expertise in specific areas
Credibility	I am comfortable accepting suggestions from my team members
	I trust that other team members' knowledge is credible
	I am confident relying on information that my team members provide to a discussion
	When my team members provide information, I want to double-check it for myself (reversed)
	I do not have much faith in my team members' expertise (reversed)
Coordination	My team members and I work together in a well-coordinated fashion
	My team members and I have very few misunderstandings when completing a task
	My team members and I complete a task on the first attempt
	My team members and I accomplish a task smoothly and efficiently (reversed)
	There is not much confusion when my team members and I set out to accomplish a task (reversed)

Figure 1. TMS 15 items scale according to Lewis (2003)

indicate their level of expertise by entering their grade obtained in their Java programming module, their ID and level of study. All data was treated as confidential and anonymous.

Measuring TMS Level

TMS level for each group A and B was measured using the 15-items scale questionnaire from Lewis (2003). As shown in **Figure 1**, the questionnaire comprises three sections namely *specialization*: the extent to which team members have specialized knowledge, *credibility*: the extent to which team members believe that the knowledge of their teammates is credible, and *coordination*: the extent to which team members are able to work collectively and access each other's expertise. Each of the TMS scales when scored on a 5-point Likert type scale ranging from 1 (strongly agree) to 5 (strongly disagree) could be used to obtain an overall estimation of the level of TMS within a team.

Independent, Dependent and Controlled Variables

Independent variables are unaffected by other variables that are being measured. In this study, two independent variables are identified: *TMS* and *Problem Complexity*. TMS was measured using the scale proposed by Lewis (2003) and *Problem Complexity* was measured according to the task complexity model developed by Tran-Cao et al. (2004). Two software-related problems, P_{Low} and P_{High}, were devised such that following complexity calculations proposed by Tran-Cao et al., one problem, P_{Low} was identified as having a *low* complexity whereas the other problem, P_{High} had a *high* complexity. Each problem had to be solved using the Java programming language and were within the scope of the syllabus covered by participants recruited.

Dependent variables are expected to vary as a consequence of an experimental manipulation of the independent variables. In this experiment, the dependent variable of interest is *Team Performance.* We objectively measure team performance using the sum of time taken to complete the given software-related problems P_{High} and P_{Low} and the number of errors found in the corresponding solution.

Controlled variables need to be kept constant so that the effect observed in an experiment is actually due to manipulation of the independent variables. As discussed earlier, there are several factors that can influence team performance. To ensure correct experimental observations, factors such as that team size, programming skills, programming environment and tool, and team diversity (age group, culture, language, etc.) are kept similar for both groups A and B.

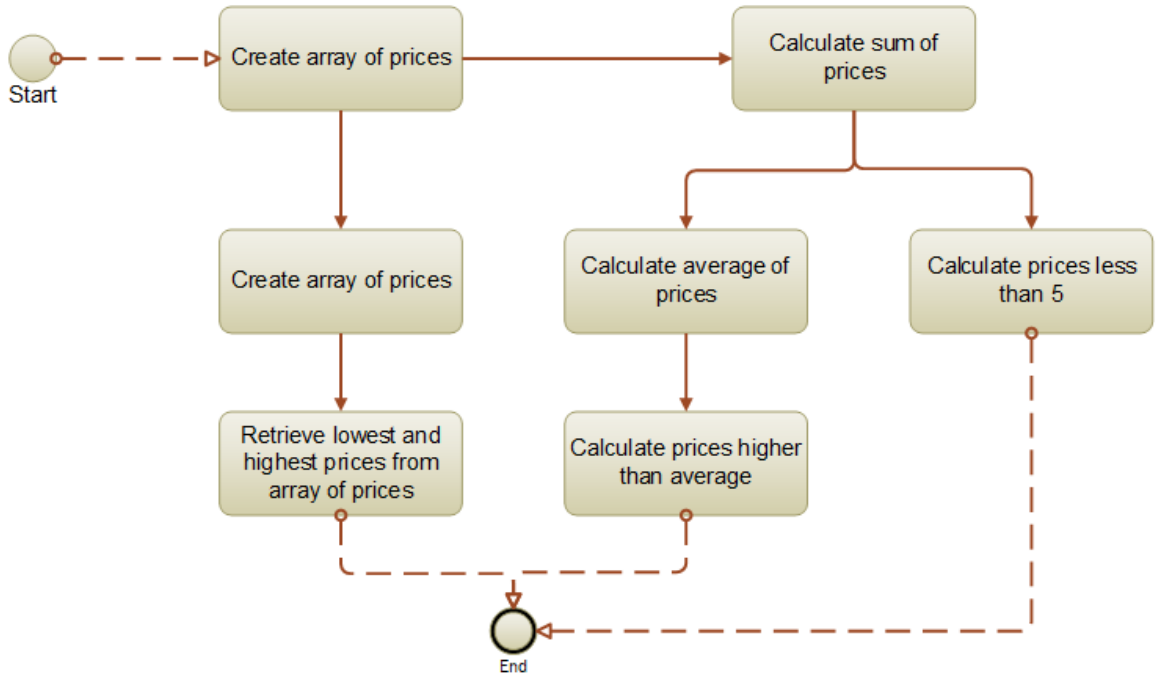

Figure 2. Functional specifications for P_{Low}

CONDUCTING THE EXPERIMENT

Prior to the experiment, a pretest exercise was conducted with six subjects to determine the time needed to complete the given problems set for the study and also to uncover any issue with the experimental design. It was determined that each problem, P_{High} and P_{Low}, could normally be completed within about one hour and that there were no major issues regarding observations or data collection.

Team Distribution and TMS Evaluation

Following advertisement, a total of 80 participants showed interest in the study. All participants had to fill in an online form set up for the experiment and each participant provided details such as their ID, score obtained in their Java programming class, contact email and level of study. Participants of similar grades in Java programming were contacted and 66 participants were retained and randomly assigned to teams of 3 individuals such as 22 teams were obtained.

Each participant was then informed about their respective team identified by a team ID. The participants were given a research consent form to sign. Each team was also convened and given a short programming exercise to solve. This exercise was essential so as to enable team members to get acquainted of each member's expertise. Once the programming exercise was completed, team members were asked to fill in the TMS questionnaire of Lewis (2003) individually. Team members indicated their team IDs on their respective TMS questionnaire, which was then analysed.

Scores obtained from the team questionnaires allowed for an indication of the TMS level of each team. As proposed by Lewis (2003), range of scores along with interval scores could be used to differentiate between a high and a low TMS. We observed that 10 teams fell in the low TMS category while the remaining 12 teams could be categorized as having a high TMS. To ensure balanced groups for the experiment, only two groups of 10 teams were retained and distributed according to the experimental set up shown in **Table 1** earlier.

Experimental Procedure

The experiment took place in a computer lab and each computer was equipped with the same Java development environment. Participants were grouped in their respective teams as discussed earlier and all participants were briefed about the experiment. Each team was then given the specifications for P_{Low} to solve, and upon completion, they were given the specifications for P_{High} to solve. **Figures 2** and **3** give an overview of the different functional requirements for P_{Low} and P_{High}. For each problem and team, the start and end time was recorded. Solutions to the two problems were saved and collected for further analysis once all teams finished solving the two problems. It is to be noted here that due to different availabilities of team members, several sessions had to be conducted for collecting data from all 20 teams.

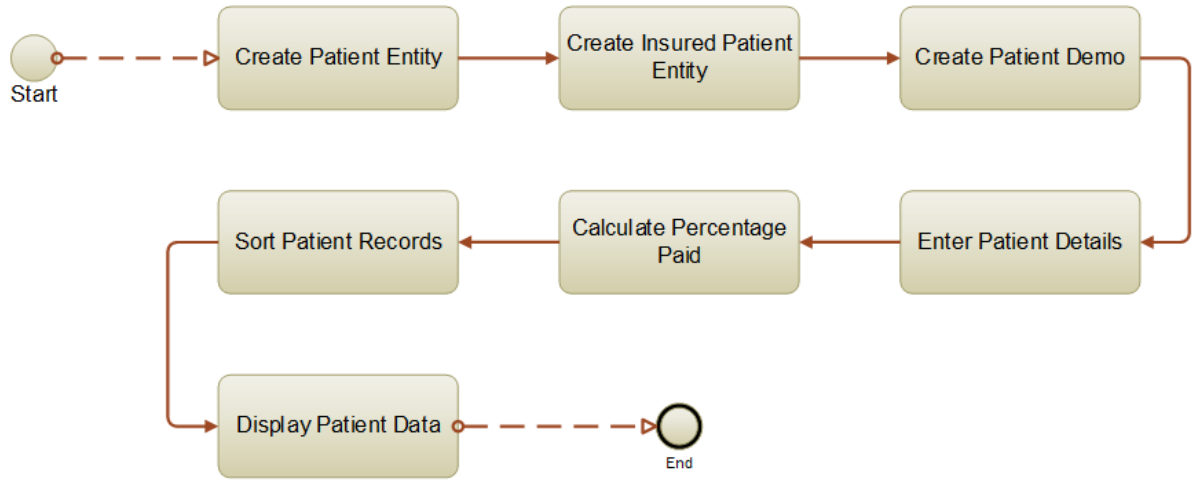

Figure 3. Functional specifications for P_{High}

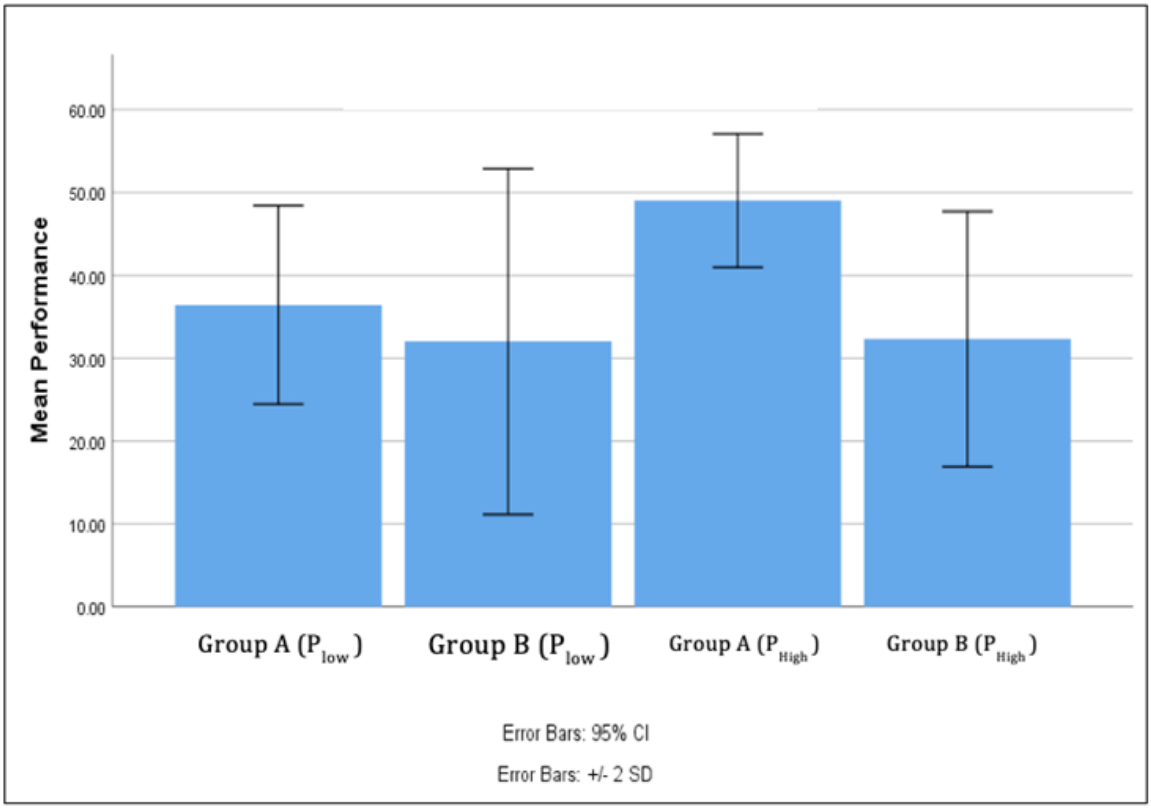

Figure 4. Mean Team Performance obtained for each group with standard error bar

DATA ANALYSIS AND RESULTS

A spreadsheet was used to record the time taken to complete a problem for each team and two independent scorers analysed the responses provided for the two problems and assigned a performance score for each team. The performance score was calculated by adding the time taken to complete a problem with the number of errors found in each corresponding solution. For data analysis, the statistical software SPSS, version 25 was used to conduct a two-way ANOVA on the data obtained. Two-way ANOVA is particularly useful to determine the effect of two independent nominal variables on a continuous variable. We also make the assumptions that there is independence between samples drawn for the experiment as a result of random assignment of participants to different groups, with same variance between data in the different groups following care taken in participant recruitment process to select subjects who exhibited similar programming performances and that each sample is taken from a normal distribution as per mean performance distribution shown in **Figure 4**.

Figure 4 shows the mean team performance observed for each group with a higher mean value indicating poorer performance, i.e., more time taken to complete a task was noted and/or more errors was found in the solution provided. To determine whether

Table 2. Comparing mean team performance obtained for LOW complexity problem (low versus high TMS)

	Sum of Squares	Df	Mean Square	F	Sig
Between Groups	80.000	1	80.000	1.685	0.211

Table 3. Comparing mean team performance obtained for HIGH complexity problem (low versus high TMS)

	Sum of Squares	Df	Mean Square	F	Sig
Between Groups	296.450	1	296.450	6.349	0.021

Table 4. Comparing mean team performance obtained for Low TMS (low versus high problem complexity)

	Sum of Squares	Df	Mean Square	F	Sig
Between Groups	84.050	1	84.050	1.414	0.250

Table 5. Comparing mean team performance obtained for High TMS (low versus high problem complexity)

	Sum of Squares	Df	Mean Square	F	Sig
Between Groups	785.250	1	785.250	30.85	0.000

any of the differences noted among the performances occurred by chance, we statistically evaluate each of our hypotheses individually for simple main effects and then we investigate for any interaction effect.

Simple Main Effect of TMS on Team Performance for Low Complexity Problem

Here we statistically compare the mean team performance of Group A (P_{Low}) against that of Group B (P_{Low}). We recall that Group A consisted of 10 teams of 3 individuals who exhibited low TMS whereas Group B consisted of 10 teams of 3 individuals who exhibited high TMS. **Table 2** shows the results obtained when conducting a two-way ANOVA analysis on the data collected.

Considering the effect of TMS when the problem at hand is of low complexity, we do not find sufficient evidence to reject the null hypothesis, H_{A0}. As indicated in **Table 2**, no significant effect was observed ($p > 0.05$). Therefore, we interpret the results as follows: When software development teams work on a low complexity problem, TMS seems to have no effect on team's performance.

Simple Main Effect of TMS on Team Performance for High Complexity Problem

In regards to the effect of TMS when the problem at hand is of high complexity, we find sufficient evidence to reject the NULL hypothesis and accept the alternate hypothesis, H_{A1} ($p < 0.05$). ANOVA results are shown in **Table 3**. Therefore, we find sufficient evidence to state that when software development teams work on a high complexity problem, TMS is likely to have an effect on team's performance.

Simple Main Effect of Problem Complexity on Team Performance with Low TMS

To understand whether there is any significant difference between team performances with low TMS when working on problems with different complexities, we proceeded in comparing the mean performance of Group A and Group B separately. **Table 4** shows the results obtained when comparing team performance for Group A (P_{Low}) against team performance for Group A (P_{High}).

Interestingly, we observe that no significant difference between team performances for Group A (Low TMS) when addressing a problem with low complexity compared to addressing a problem with high complexity ($p > 0.05$). Consequently, we could not obtain sufficient evidence to reject the null hypothesis, H_{A0}. Therefore, software development teams with a low TMS are likely to perform equally regardless of the complexity of the problems at hand. In our experiment, we noted a relatively poorer performance for teams with lower TMS than teams with higher TMS.

Simple Main Effect of Problem Complexity on Team Performance with High TMS

Table 5 shows the results obtained when comparing team performance for Group B (P_{Low}) against team performance for Group B (P_{High}) where Group B consist of teams exhibiting high TMS.

Given that $p < 0.05$, we obtain sufficient evidence to reject the Null hypothesis and accept the alternate hypothesis H_{A1}. Therefore, we determine that software development teams with a high TMS will more likely have different performance when addressing problems of different complexities.

Interaction Effect of Problem Complexity and TMS on Team Performance

To determine whether task complexity may interact with TMS when interpreting results for team performance, we proceeded in evaluating for any interaction effect. As per **Table 6**, we observe a significant interaction effect between the two dependent variables, TMS and problem complexity ($p < 0.05$).

Consequently, it is understood that the effect of TMS on team performance will also depend on the problem complexity. We discuss the results obtained further.

Table 6. Evaluating interaction effect between TMS and problem complexity

	Sum of Squares	Df	Mean Square	F	Sig
TMS Level *vs.* Problem Complexity	198.025	1	198.025	4.716	0.037

RESULTS INTERPRETATIONS, DISCUSSION AND IMPLICATIONS

In this study, we sought to determine whether there was any relationship between TMS and team performance. We added another dependent variable, namely, problem complexity in the equation since we believed that problems with different complexities could require different level of expertise that could further demonstrate the importance of TMS in a team. In general, we observe that there is no significant effect of TMS on team performance between the different groups, except for when problem complexity is high and when a group exhibits a high TMS.

In other words, teams with high TMS have a tendency to perform better especially when the problem at hand is complex. One possible reason to explain the observed significant effect is that in a high complexity problem, interdependence on members is higher as team members often need to perform a number of tasks, which involves input and collaboration from individuals with different expertise. When TMS is high, members already know about their respective expectations and deliverables within the team and are able to perform task activities more efficiently. In contrast, in a low TMS team, members are unaware of other member's abilities and are thus unable to channel efforts towards a good team performance. Results observed here are consistent with previous study conducted by Mohamed et al. (2013), who reported that a high TMS is more likely to yield better team performance and with Li et al. (2015), who asserted that a low TMS negatively affects to team performance. When the problem at hand is of low complexity, it seems unlikely that TMS will have any have effect on team performance regardless of the TMS level. One possible reason to explain why no significant effect of TMS level on team performance was observed could be due to lower reliance on team specific expertise, which may not necessary be considered crucial to address a given problem.

We also retain that for teams with low TMS, no significant effect was observed on team performance when problem complexity was varied. However, for teams with high TMS, when problem complexity was varied from low to high, a significant effect was observed. A possible reason to explain why teams with low TMS do not exhibit any change in team performance could be due to lack of awareness of team members abilities and also due to low level of interaction or communication between the members. Low TMS teams is an indication that team members are not necessary acquainted with other members and such condition could be detrimental to team performance as it creates a dysfunctional block of skilled persons instead of a collaborative one. In contrast, a high TMS in a team can be interpreted as a highly cohesive team functioning as one group towards the same goal such that when required, as in the case of a high complexity problem, team members are able to communicate accordingly and pull available resources together to solve a given problem.

Findings presented in this research provide several important implications for software development team within industrial practices. Firstly, this study provides sufficient indication to encourage project managers to consider when setting up teams. Given that software projects are usually complex, it follows that TMS should not be overlooked when setting up teams. Secondly, given the possible effect of TMS on team performance, organizations may wish to develop strategies within their staff training programs to include development of TMS as well. When team members are trained together, they develop a shared understanding toward expertise required to accomplish tasks effectively. Thus, training groups of individuals instead of focusing on individuals could help teams build appropriate mental models about other team members and increase team performance. Thirdly, software development professionals work in a dynamic environment whereby new skills and expertise are gained continuously. Consequently, it is expected that TMS within teams will also be dynamic. TMS in teams must be subjected to update as well and this could be achieved by encouraging frequent interaction among team members. To do so, software project managers are encouraged to carry out activities such as group coaching, job rotation, collective decision-making, feedback sharing, among other team oriented activities.

CONCLUSION

From our experience with the software development industry and from findings reported in past studies, we have noted that consideration for TMS in setting up software development team is practically inexistent or is overlooked. This study is expected to shed light to the emerging body of literature on transactive memory systems or TMS on team performance when developing software as a team. We find that in general TMS will not have any effect on a team performance when the problem at hand is fairly simple. However, when the problem to be addressed is complex, a team with high TMS is more likely to have a better performance than a team with lower TMS. We have also presented a means to measure TMS that we consider useful for project managers to adopt when setting up teams. TMS should, however, not be regarded as a single factor to determine team performance. Other factors such as creativity, innovation or motivation of team members are also important as demonstrated by numerous studies on the topic. As future work, the interaction between different factors that influence software performance can be evaluated. In addition, since in this experiment, we relied on undergraduates as subjects who are familiar with Java as a programming language and the problems given do not reflect real world settings, where projects can be more elaborated and spanning several weeks or months, we anticipate that more accurate results will be obtained if the same experiment is repeated in a work setting, on a larger scale and with other programming languages. We also used a simplistic approach (time taken to complete a task and number of

errors found in final codes) to measure team performance. Such a measure helped in uniformly comparing the final products across different groups. As future work, the performance of each team could also be measured according to the quality of codes produced for a more industry focused evaluation of results observed.

REFERENCES

Acuña, S. T., Gómez, M. and Juristo, N. (2008). Towards understanding the relationship between team climate and software quality—a quasi-experimental study. *Empirical software engineering*, *13*(4), 401. https://doi.org/10.1007/s10664-008-9074-8

Ahn, J., DeAngelis, D. and Barber, S. (2007). November. Attitude driven team formation using multi-dimensional trust. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*. IEEE, pp. 229-235. https://doi.org/10.1109/IAT.2007.77

Akgün, A. E. (2020). Team wisdom in software development projects and its impact on project performance. *International Journal of Information Management, 50*, 228-243. https://doi.org/10.1016/j.ijinfomgt.2019.05.019

Akgun, A. E., Byrne, J. C., Keskin, H. and Lynn, G. S. (2006). Transactive Memory System in New Product Development Teams. *IEEE Transactions on Engineering Management, 53*(1), 95-111. https://doi.org/10.1109/TEM.2005.857570

Beaver, J. M. and Schiavone, G. A. (2006). The effects of development team skill on software product quality. *ACM SIGSOFT Software Engineering Notes*, *31*(3), 1-5. https://doi.org/10.1145/1127878.1127882

Blasi, L., Fiore, S. M., Hedberg, J. and Schmid, R. F. (2008). The Social consequences of design and development teams. In: J. M. Spector, M. D. Merrill, J. Van MerriÎnboer and M. P. Driscoll (eds.). *Handbook of Research on Educational Communications and Technology* (pp. 647-658). New York: Lawrence Erlbaum Associates.

Cohen, S. G. and Bailey, D. E. (1997). What makes teams work: Group effectiveness research from the shop floor to the executive suite. *Journal of Management, 23*, 239-290. https://doi.org/10.1177/014920639702300303

da Silva, F. Q., França, A. C. C., Suassuna, M., de Sousa Mariz, L. M., Rossiley, I., de Miranda, R. C., Gouveia, T. B., Monteiro, C. V., Lucena, E., Cardozo, E. S. and Espindola, E. (2013). Team building criteria in software projects: A mix-method replicated study. *Information and Software Technology*, *55*(7), 1316-1340. https://doi.org/10.1016/j.infsof.2012.11.006

Dai, Y., Du, K., Byun, G. and Zhu, X. (2017). Ambidexterity in new ventures: The impact of new product development alliances and transactive memory systems. *Journal of Business Research*, *75*, 77-85. https://doi.org/10.1016/j.jbusres.2017.02.009

Dhir, S., Kumar, D. and Singh, V. B. (2019). Success and failure factors that impact on project implementation using agile software development methodology. In *Software Engineering*, *Springer, Singapore*, pp. 647-654. https://doi.org/10.1007/978-981-10-8848-3_62

Ebert, C. and De Neve, P. (2001). Surviving global software development. *IEEE software*, *18*(2), 62-69. https://doi.org/10.1109/52.914748

Faraj, S. and Sproull, L. (2000). Coordinating expertise in software development teams. *Management Science, 46*(12), 1554-1568. https://doi.org/10.1287/mnsc.46.12.1554.12072

França, A.C.C., Lucena, É.F. and da Silva, F.Q., 2009, November. A quantitative assessment on team building criteria for software project teams. In *Proceedings of 6th Experimental Software Engineering Latin American Workshop (ESELAW 2009)* (p. 12).

Fuggetta, A. (2000). Software process: a roadmap. Limerick, Ireland, *ICSE '00 Proceedings of the Conference on The Future of Software Engineering*. https://doi.org/10.1145/336512.336521

Gorla, N. and Lam, Y. W. (2004). Who should work with whom?: building effective software project teams. *Communications of the ACM, 47*(6), 79-82. https://doi.org/10.1145/990680.990684

He, J., Butler, B. S. and King, W. R. (2007). Team cognition: Development and evolution in software project teams. *Journal of Management Information Systems*, *24*(2), 261-292. https://doi.org/10.2753/MIS0742-1222240210

Hoegl, M. and Gemuenden, H. G. (2001). Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept. *Organization Science, 12*(4), 435-449. https://doi.org/10.1287/orsc.12.4.435.10635

Katzenbach, J. R. and Smith, D. K. (1993). *The wisdom of teams: Creating the High-performance Organization*. Harvard Business School Press.

Kozlowski, S. W. J. and Klein, K. J. (2000). A multilevel approach to theory and research in organizations: Contextual, temporal, and emergent processes. In: S. W. J. Kozlowski and K. J. Klein (eds.). *Multilevel theory, research, and methods in organizations: Foundations, extensions, and new directions* (pp. 3-90). San Francisco, CA, US: Jossey-Bass.

Laughery Jr, K. R. and Laughery Sr, K. R. (1985). Human factors in software engineering: A review of the literature. *Journal of Systems and Software*, *5*(1), 3-14. https://doi.org/10.1016/0164-1212(85)90003-2

Lewis, K. (2003). Measuring transactive memory systems in the field: Scale development and validation. *Journal of Applied Psychology, 88*(4), 587-604. https://doi.org/10.1037/0021-9010.88.4.587

Li, Y.-J., Hao, S.-Y. and Ren, X. (2015). The Effect of Transactive Memory Systems on Team Performance: The Mediating Role of Knowledge Sharing and the Moderating Role of Task Complex. *International Conference on Management Science & Engineering, Dubai, United Arab Emirates*.

Mohamed Ariff, M. I., Sharma, R., Milton, S. and Bosua, R. (2013). Modeling the effect of task interdependence on the relationship between transactive memory systems (TMS) quality and team performance. Kuala Lumpur, Malaysia, *2013 International Conference on Research and Innovation in Information Systems (ICRIIS)*. https://doi.org/10.1109/ICRIIS.2013.6716679

Moreland, R. L. and Myaskovsky, L. (2000). Exploring the Performance Benefits of Group Training: Transactive Memory or Improved Communication?. *Organizational Behavior and Human Decision Processes, 82*(1), 117-133. https://doi.org/10.1006/obhd.2000.2891

O'Connor, R. V. (2008). Human Aspects of Information Technology Development. *International Journal of Technology, Policy and Management, 8*(1).

Ong, A. and Kankanhalli, A. (2005). Team Expertise and Performance in Information Systems Development Projects. *Pacific Asia Conference on Information Systems,* Bangkok.

Oztaysi, B., Onar, S. C. and Kahraman, C. (2019). Performance Measurement Model for Software Development Teams Using Interval-valued Intuitionistic Fuzzy Analytic Hierarchy Process. *Journal of Multiple-Valued Logic & Soft Computing*, *33*(4/5), 321-329.

Rapp, T. L. and Mathieu, J. E. (2019). Team and individual influences on members' identification and performance per membership in multiple team membership arrangements. *Journal of Applied Psychology*, *104*(3), 303. https://doi.org/10.1037/apl0000344

Rulke, D. L. and Rau, D. (2000). Investigating the encoding process of transactive memory development in group training. *Group & Organization Management*, *25*(4), 373-396. https://doi.org/10.1177/1059601100254004

Ryan, S. and Connor, R. (2019). Team Tacit Knowledge as a Predictor of Performance in Software Development Teams. *Mind*, *3*, 31.

Salas, E., Cooke, N. J. and Rosen, M. A. (2008). On Teams, Teamwork, and Team Performance: Discoveries and Developments. *Human Factors the Journal of the Human Factors and Ergonomics Society, 50*(3), 540-547. https://doi.org/10.1518/001872008X288457

Schnabel, I. and Pizka, M. (2006). Goal-Driven Software Development. Columbia, MD, USA, *30th Annual IEEE / NASA Software Engineering Workshop*. https://doi.org/10.1109/SEW.2006.21

Smith-Jentsch, K. A., Campbel, G. E., Milanovich, D. M. and Reynolds, A. M. (2001). Measuring Teamwork Mental Models to Support Training Needs Assessment, Development, and Evaluation: Two Empirical Studies. *Journal of Organizational Behavior, 22*(2), 179-194. https://doi.org/10.1002/job.88

Sommerville, I. (2019). *Engineering Software Products*. London, UK: Pearson.

Sudhakar, G. P., Farooq, A. and Patnaik, S. (2011). Soft factors affecting the performance of software development teams. *Team Performance Management, 17*(3/4), 187-205. https://doi.org/10.1108/13527591111143718

Terricone, P. and Luca, J. (2002). Employees, teamwork and social interdependence-a formula for successful business? *Team Performance Management: An International Journal, 8*(3/4), 54-59. https://doi.org/10.1108/13527590210433348

Tran-Cao, D., LÈvesque, G. and Meunier, J.-G. (2004). Software functional complexity measurement. In *Proceedings of the International Conference on RIVFí04, Hanoi, Vietnam*.

Trendowicz, A., Münch, J. and Jeffery, R. (2008). October. State of the practice in software effort estimation: a survey and literature review. In *IFIP Central and East European Conference on Software Engineering Techniques*. Springer, Berlin, Heidelberg, pp. 232-245. https://doi.org/10.1007/978-3-642-22386-0_18

Tsai, Y. H., Joe, S. W., Chen, M. L., Lin, C. P., Ma, H. C. and Du, J. W. (2016). Assessing team performance: Moderating roles of transactive memory, hypercompetition, and emotional regulation. *Human Performance*, *29*(2), 89-105. https://doi.org/10.1080/08959285.2016.1154059

Walz, D. B., Elam, J. J. and Curtis, B. (1993). Inside a software design team: knowledge acquisition, sharing, and integration. *Communications of the ACM, 36*(10), 63-77. https://doi.org/10.1145/163430.163447

Wang, Y., Huang, Q., Davison, R. M. and Yang, F. (2018). Effect of transactive memory systems on team performance mediated by knowledge transfer. *International Journal of Information Management*, *41*, 65-79. https://doi.org/10.1016/j.ijinfomgt.2018.04.001

Wegner, D. M. (1986). Transactive Memory: A Contemporary Analysis of the Group Mind. In: B. Mullen and G. Goethals (eds.), Theories of Group Behavior. New York, NY: Springer Series in Social Psychology, pp. 185- 208. https://doi.org/10.1007/978-1-4612-4634-3_9

Zhang, Z. X., Hempel, P. S., Han, Y. L. and Tjosvold, D. (2007). Transactive memory system links work team characteristics and performance. *Journal of applied psychology*, *92*(6), 1722. https://doi.org/10.1037/0021-9010.92.6.1722