Improving Knowledge Sharing in Distributed Software Development

Sara Waheed[1] , Bushra Hamid [2], NZ Jhanjhi[3], Mamoona Humayun[4], Nazir A Malik[5]
Department of Information Technology, PMAS Arid Agriculture University, Rawalpindi, Pakistan[1, 2]
School of Computing & IT (SoCIT), Taylor's University, Subang Jaya, Selangor, Malaysia[3]
College of Computer and Information Science (CCIS), Jouf University Al-Jouf, Saudi Arabia[4]
Department of Computer Science, Bahria University, Islamabad, Pakistan[5]

***Abstract*—Distributed Software Development has become an established software development paradigm that provides several advantages but it presents significant challenges to share and understand the knowledge required for developing software. Organizations are expected to implement appropriate practices to address knowledge management. From the existing studies, it is been analyzed that there were problems of collaboration between distributed team members which effects knowledge sharing. Documentation problem (such as missing, poor and outdated documents) and knowledge vaporization (as much of the conversation and communication is done via chat and retrieving it later is a great headache) is a major challenge in Distributed Software Development in knowledge sharing. Our main objective is to improve knowledge sharing between distributed team members and prevent knowledge vaporization and reduced documentation problem that will help in improving software development process in a distributed environment. To eliminate these challenges we proposed a framework which deals with documentation and knowledge vaporization problems and evaluated it through industrial case study and evaluate the framework performance in real-life context where actually the problem arises, we conducted the interviews and analyzed the data using thematic analysis and SUS questioner we came to the conclusion on team members response that they are satisfied with our proposed solution and it improved their knowledge sharing process. Our intention was to improve the knowledge process with our proposed solution and the evaluation showed that we resolved these problems.**

***Keywords*—Distributed software development; knowledge sharing; knowledge management**

I. INTRODUCTION

From few decades' creation maintenances and development of software become advanced from being centralized (at one location) to being dispersed at several locations [1], in distributed development teams are scattered geographically at multiple sites while working on the same product this concept is known as Distributed Software Development (DSD). Multiple sites include a different location such as different cities in the same country or it may be scattered along with the globe. DSD offers numerous benefits such as intense resource pool, reduction in cost, less resource consumption, variety of different skills and expertise around the world and continuous working around the clock. These benefits overall increase the quality of the software products [29]. Along with the several advantages of DSD, it brings numerous challenges that are geographically distributed teams may encounter such as product quality compromise due to team dispersion, Coordination, Communication and Collaboration challenges, lack of face to face communication leads to non-trust worthy behavior [2] and not sharing required knowledge. Many techniques support DSD approach and literature presented numerous ways to tackle these challenges but still, there is a need of enhancement in some areas such as knowledge management many existing techniques address knowledge sharing but it lacks to some extent.

Knowledge management is considered to be the most required resource of an organization [1]. Knowledge can be explained as "Knowledge, while made up of data and information, can be thought of as much greater understanding of a situation, relationships, causal phenomena, and the theories and rules (explicit and implicit both) that underlie a given domain or problem" [30]. Knowledge management is considered to be a very vast field, it provides ways to share knowledge and aids in increasing mutual understanding solves collaboration and coordination challenges [3]. In an organization lots of knowledge resides in different software processes, activates, organizational assets and methodologies, environment, knowledge reside in team members mind. It is very important to share and transfer the knowledge to deliver the product to the customer which he/she requested knowledge is needed to be managed shared and transferred from the beginning to the end of SDLC (software development life cycle) [4].

When the teams are geographically distributed they need to share knowledge in explicit form for that documentation plays an important role but in distributed development, there is a lack of proper documentation [5], [6]. Most of the organization in these days are using agile approaches which does not support many docsumentation [7], [8] and they also focus on sharing tacit knowledge there is a lack of creating and maintaining explicit knowledge much of the product knowledge remains in source code, test files, documentation remains outdated as regard to the project the dispersed team members need proper documentation for understanding the product knowledge incomplete and abstract documents are not enough for effective knowledge sharing from the literature documentation problems (such as poor, missing and outdated documentation) is identified [5], [6], [9]. Also agile and distributed development often clashes due to their distant nature.

Alongside documentation problem another issue is knowledge vaporization local team members and remote team members need to communicate to work together on the same project much of the knowledge is existing in electronic media such as during chat retrieving this knowledge is not easy because it's not easily accessible at one place [7], [8], [10]. So our main focus is to reduce these issues to facilitate knowledge sharing in distributed development. In this paper, our main objective is to improve knowledge sharing between distributed team members and prevent documentation problem and knowledge vaporization it will help in improving Software Development Process in a distributed environment and help to get more advantages of DSD. Therefore our study aimed to explore the following research questions:

RQ1: What are the existing knowledge exchange/sharing mitigation strategies in distributed development?

RQ2: What are the knowledge exchange/sharing issues and challenges in distributed software development?

RQ3: Does the proposed solution overcome the knowledge vaporization and documentation problem in distributed software development?

In the next sections we will present a literature review and then we will propose our solution and we will present its evaluation and results followed by limitation, future work and conclustion of the study.

II. Literature Review

In DSD multiple sites are involved in the development of a certain product, DSD does not necessarily require multiple number of organization, there can be one organization with different branches that can be located in different cities either in one country or in different countries when the organizations are from around the globe there is a time zone difference as well as cultural diversity arises because one organization share the same culture where as multiple organizations do not share the same culture, traditions and languages. So DSD provides versatility in team members. In DSD the situation becomes more problematic when the multiple team members from different culture, language, time zone, and geographical locations works at the same project [1], [2] communications and collaboration among team members become a challenge and knowledge sharing in this scenario seems totally impossible.

In our research, our main focus is on knowledge sharing when the teams are geographically distributed. The knowledge management process includes knowledge creation, knowledge store and retrieval, knowledge sharing and knowledge application. Knowledge is categorized into two different levels explicit knowledge and tacit knowledge. The Explicit type of knowledge is in countable form like a written document, books, any material which is in physical form, explicit knowledge is easy to transfer, whereas tacit knowledge is the knowledge resides in peoples mind, they can include skills, thoughts, ideas, perceptions, values, and faiths so it is very hard to share this type of knowledge [1], [11]. Knowledge sharing is the process to transfer the knowledge of source to the destination. The source and destination can be anything like individual groups within the same company or different companies where the team members are scattered [12]. Knowledge sharing is very essential for project success and for teams to work together. And it is one of the key domain being affected by the DSD. As the base of developing software relies on sharing knowledge, and whose success factor is wholly based on practical sharing of knowledge around software developers of distributed teams. The most complicated phase is to share both explicit and tacit knowledge among the geographically distributed teams. In study [13], [14], knowledge sharing is considered to be the essential process of knowledge management. In DSD vendors usually have the knowledge and technical expertise of a project, while the clients hold the requisite and application domain knowledge. If the clients and vendors are unable to share the product knowledge, the clients may not suitably supply requirements related to business and products domain so in the absence of knowledge understating of the vendors without proper domain knowledge causes negative effect on product development and they cannot effectively and efficiently use their skills and technical expertise to build the product effectively [15].

The use of approaches in Agile and importance on tacit communication may perhaps negatively affect creating and maintaining knowledge which is explicit [16], [17]. According to studies [5], [17] describes much of the project knowledge remained scattered in test cases and source code, documents are not synced with the required and updated information which is the reason for misunderstanding between distributed teams who are looking for accurate knowledge. These studies [5], [6], [16], [17], [18], [19], [20], [21] shed light on the effect of the absence of proper and consistent documentation on knowledge sharing in a distributed environment. Abstract requirement description was not sufficient for offshore teams it's another reason for causing misunderstanding [21], [22]. Study [21] Identified that requirements are written informally on personal notebooks and whiteboards; this method found considered improper and forbid knowledge transfer properly because it was difficult for teams in offshore to make social relations with the users in business [21]. Also most of the companies in today's era are using agile methodology for software development, agile focuses more on short iteration and source code and less attention is paid to documents [7], [8], whereas in DSD where teams are geographically distributed they need to share the knowledge on a daily basis for that they use synchronous and asynchronous means for communication. Most of the communication is done via chats, emails, video conferencing there is major issue of knowledge vaporization because in many informal chats some important conversations happen which becomes difficult to remember at the time of need or when we want to retrieve it. Knowledge vaporization is a major challenge because much of the knowledge is available in unstructured electronic media retrieving this knowledge is not easy because it's not easily accessible and a time consuming process [7], [8], [20], [23], [24], [25]. There are some knowledge sharing challenges and their mitigation strategies are identified from the literature amd its answers RQ1 and RQ2 [6], [9] and presented in Table I.

TABLE. I. KNOWLEDGE SHARING CHALLENGES AND APPROACHES

Knowledge Sharing challenges in Distributed Software Development	Knowledge Sharing Approaches in Distributed Software Development
Ch1: Knowledge sharing expenditure Ch2: Workers turnover rate Ch3: Lower priority recognition to sharing knowledge activities Ch4: Structural hierarchy Ch5: Ambiguous characterization of roles and responsibilities Ch6: Problems in documentation Ch7: The difference of contextual settings Ch8: Variation in educational and technical expertise Ch9: Social impediments, contextual and social impediments Ch10: Technological and Organizational impediments Ch11: Knowledge vaporization Ch12: Flaws in retaining group awareness Ch13: Technology and cultural impediments Ch14: Knowledge storing issue Ch15: Lack of knowledge awareness mechanisms Ch16: Communication barriers due to distance	AP1: Bonus and motivation AP2: Short term collocation AP3: Flexible communication structure AP4: Understandable work-structure AP5: Combined work between sites AP6: The documentation problem AP7: Confirming common understanding between sites AP8: The usage of boundary spanning roles AP9: Forming virtual communities of practice AP10: Continuous knowledge transfer process in the organizations AP11: Knowledge sharing in agile virtual teams AP12: Providing groupware AP13: Success model for knowledge management AP14: Efficient storage of knowledge AP15: Novel Expertise and solutions AP16: Raising team qualification and expertise

III. PROPOSED FRAMEWORK

The framework is presented in Fig. 1. In our research we provide a solution for poor, missing, outdated documentation problem and knowledge vaporization in dispersed teams. Our framework deals with poor documentation problem by creating documents artefacts(product specification, design specification, development handbook, etc.), missing documentation is facilitated by identifying related documents of the projects, outdated documentation facilitated by the syncing mechanism so that every team member have the latest document at their workstation, knowledge vaporization is facilitated by tagging mechanism following are the main phases of framework.

A. Actors Layer

Actor's layer characterizes the arrangement of the distributed teams and interactions. In DSD there are onshore and offshore companies working together on a product, developers and managers act as the main actors for sharing the artefacts. These artefacts are linked with the actors as there is a client or vendor team working in site A and their offshore vendor team working at site B. So the developers and managers at site A use their own product specification and design specification artefacts and share them because they are colocated and they do not need to share with the vendor team at site B, Then the managers at site A prepare product specification and design specification artefacts for vendor team developers and managers at site B which is specifically prepared for them and shared between developer and managers at site B and manager of site A. Information architecture and development handbook artefacts shared between remote location's developers and manager.

B. Artefacts Layer

To deal with the poor documentation artefact based knowl edge sharing technique will resolve the problem of poor documentation. Software Engineering is typically assisted by a broad diversity of artefacts that documents numerous knowledge types concerning to the software product being created or maintained. Software documentation plays a very vital role in sharing of knowledge especially when the teams are geographically distributed and there is no way of face to face interaction opportunity so in this case, the teams rely heavily on documentation to develop a common understanding about the product being developed. As agile practices are becoming famous and adopted by many firms agile practices promote and focuses on creating executables artefacts such as source code rather than producing artefact that documents the knowledge about requirements, architecture and design decisions. We classified the artefacts as architectural information specification document, product specification document, and design specification document and development standards handbook. Summary of these documents is given in Table II.

TABLE. II. AN OVERVIEW OF DOCUMENT TYPES

Document Artefacts	Description
Architecture Information Specification Document	Contains snapshots drawings and sketches of flow of information with relevant reviews and comments.Represents user interface, business processes and business domain knowledge.
Product Specification Document	Contains business logics and functional requirements of the product.
Design specification Document	Contains design's detailed solution e.g. data models reports, formats, field mapping, structures of database etc
Development Standard Handbook	Contains architectural information and solutions, quality checks, naming conventions, coding standards, database design rules, tools configurations, repository configurations.

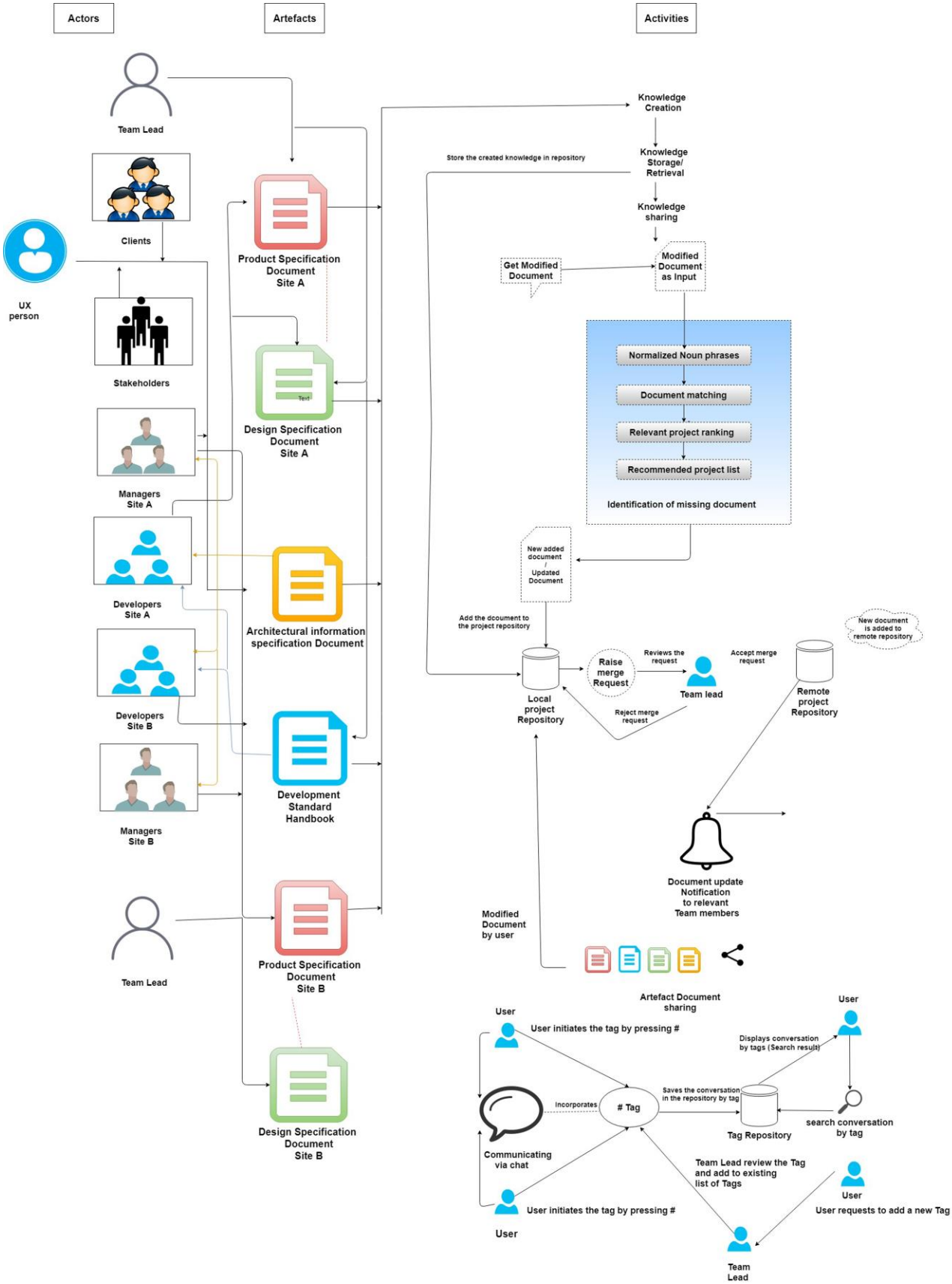

Fig. 1. Propsed Framework

C. Activity Layer

- **Knowledge Management Process:** Knowledge management process include knowledge creation, retrieval, sharing and knowledge application. Our main focus is on knowledge sharing process because the development style we choose is not central it is distributed the teams are geographically scattered at different locations they need to share the extensive amount of knowledge to develop a software product.

- **Identification of missing documents:** There is a possibility that some team member create document in a untracked directory and forgot to upload it so our framework will provide a mechanism to resolve this issue it identifies the relevant document and notifies the team member about the relevant document which may be needed for the desired project following are the steps:

1) Normalized noun phases: Noun phases will be used in the documentation as a feature for matching to a relevant project The noun phrases are annotated and normalized by applying a variety of Natural Language Processing (NLP) techniques to mine the input text and generates all relevant annotations, including sentence boundaries, tokenization, part-of-speech tags, and phrase chunking. Noun phrases are extracted based on a relatively simple pattern of part-of-speech (POS) tag sequences.

2) Document matching: The Okapi BM25 algorithm is used frequently for matching it is a technique of information retrieval. The matching document is ranked conferring to their relevance to a given document. The input will be a document which is recently modified in the system this modified document will be considered the document as input and the output contains the ranked project which mostly matches with the software project along with the scores which were based on the input document text. We converted the document text into normalizing noun phase's annotations for the matching algorithm.

3) Relevant project ranking: The Okapi BM25 can be described as below: Given an input text Q, containing noun phrases q1.qn, the BM25 score of a paper D is: Where, f (q1, d) is q1 's frequency in

$$\text{score}(D,Q)=\sum_{i=1}^{n} \text{IDF}.\frac{f\left(q_i,D\right).(k_1+1)}{f(q_i,D)+k_1.(1-b+b.\frac{|D|}{avgdl})}$$

Document D, $|D|$ is the length of document D in noun phrases, and *avgdl* is the weighted document length in the project. The parameters *ki* and *b* allow for adapting the algorithm to different use cases. In our case, used 1.5 for ki, 0.6 for b and measured 68 for *avgdl* as the average document length The frequent term like "a", "this", "an", "the text" etc. may appear in every document they are not important terms to eliminate these highly occurred terms in the document Inverse document frequency formula is used to extract non frequent terms because they are important in our case. We use IDF described below for extracting non-frequent terms Formula of Inverse document frequency is:

$$\text{IDF}\left(Q_i\right)=\log\frac{N-n\left(q_i\right)+0.5}{n\left(q_i\right)+0.5}$$

Where 'N' is the entire number of documents of the project and n(qi) is the number of documents having qi (noun phases) after this we get a ranked document list with a BM25 score for each document that already exists in the project repository the top document in the list is the document most related to the input document

4) Recommended project list: The next step is to use recommended project ranking algorithm which interprets the scores of the individual document to scores for projects. Keep all projects documents with the maximum BM25's score of the ranked documents and extract the similarity of the input document score with the existing project documents score, calculate the average BM25 score of each project by averaging the score of all documents in the project formula is:

$$\text{score}(F,Q)=\frac{\sum_{i-1}^{N^F}\text{score}(D_i,Q)}{N_P}$$

we take the average to be precise for project size. The user will get the recommended project on the basis of the matched document.

- **Document Sync:** When the document is changed by any team member it will be synced in all those team members PC's who are the relevant users a notification will be generated to let the team members know that some latest document is uploaded and they should get those update. In our case, we are using Git [26] for document versioning control and for the remote repository. When the document is matched and copied/moved to the project documentation repository. The person who is adding the document in the local repository must have to push the branch (containing the newly added document/ or updated document) to the remote repository and raise a merge request for the master branch. The responsible person (team lead) review the merge request and either accepts or rejects the request to merge the branch in the master branch if the request is accepted and user branch is merged into the master branch and the master branch got updated.

- **Document update notification:** A notification is generated when the relevant documents got updated and synced so team members will be aware of any changes made to the documents.

- **Conversation storage and retrieval by using Tagging mechanism:** To deal with knowledge vaporization can be solved with the help of tagging mechanism. The remote teams rely heavily on textual media for interaction as IM chat reduces the communication gap caused by linguistic differences. To deal with knowledge vaporization these chats interactions can be stored and retrieved later by using the tagging mechanism. There is a list of tags already available

created by some responsible and experienced team members. The user cannot add or delete the tag. The tags are added by meeting by experienced members when they think there is a need of adding or removing a tag or a meta tag, So tags are maintained by the admin panel. There are meta tags all the tags are related to this meta-tag, for example a meta tag can be #code and #javaTricks can a be tag related to the meta tag, meta tag can be explained as parent tag and each parent tag has some child tags which come under parent tag. The advantage of using Meta tag or parent tag is when the team member wants to retrieve some topic but was not able to remember the exact tag he/she can search for the parent tag initially and child tags are listed any child tag can be selected and search can be initiated. The main reason for using tagging is to restore knowledge at later times, as in distributed development teams are located at remote locations and working together on the same project. They rely heavily on using IM tools where some important knowledge existed in the team member's conversation these conversation parts can be stored by tags and these conversation act as an important reviser of knowledge. User can search the conversation via tag and gain the required knowledge when needed also if the user does not initiate the tag using # but uses the tag in the conversation the conversation will be stored because of the automatic tagging mechisam.

IV. Evaluation and Results

The evaluation of our framework was done via a case study and it answers our RQ3. It includes case study results and expert review through interviews. This section also represents the research findings and their discussion.

A. Case Study

To answer RQ3 we have taken a case study of DSD based Software Company. Our research topic is focusing on DSD so we selected a company which is distributed in nature. It is located in Islamabad, Karachi, Lahore, Rawalpindi and it's head quarter is in Karachi. The company started in 2001 and is set out to redefine Pakistan computing industry. They offers a complete range of technology services such as business applications, Managed services, and IT infrastructure and solutions. Our case study evaluation mainly focuses on to improve knowledge sharing process with the help of our proposed framework the case study used to evaluate the proposed framework's impact on knowledge sharing. In software engineering, the cases are contemporary phenomena in software engineering in real life setting. We considered development teams who are major source of knowledge sharing and where there was a need to retrieve the knowledge and the knowledge vaporization issue emerge from here also the company we selected is distributed in nature so they face documentation problem such as poor, missing and outdated documents and knowledge loss due to knowledge vaporization because the teams use chat tools for daily communication.

Our case study is a single case study and embedded in design as we had three units of analysis first unit of analysis is documentation problem and second unit of analysis is the knowledge vaporization problem in distributed teams third and the final unit of analysis is the overall case itself which is knowledge sharing in distributed environments For data collection set of semi-structured interviews were conducted one prior to explaining the framework and one after describing framework and a tool support is provided to aid the framework and its work flow was described to the team members. At the initial stage interview question were consisted of how they are currently exchanging the knowledge and how they are documenting and dealing with knowledge vaporization problem and after two weeks another interview was con ducted when the framework and tool workflow was explained and used by the development team the interview question consisted of how the framework helps them mitigate knowledge sharing challenges is there any improvement in documentation and knowledge vaporization problem. After data collection, next phase was data analysis we analyze the interview data by using qualitative data analysis method called thematic analysis.

After reviewing the interview data we performed coding on that data. Coding was performed by extracting data chunks pointing to the point where the employees describe the proposed framework impact on their work, during an exchange of knowledge via conversation and document creation and sharing. We also coded scattered quotations referring to the needs and motivation behind knowledge sharing certain practices and associated challenges. The code resulted in evolving themes in data collected. Knowledge sharing practices of team members, the extent of knowledge shared. Impact of our framework on their knowledge sharing activities, etc.

B. Framework Tool Support (Blueprint)

We provide a prototype to support our framework and its activities these are four document artefacts (as described in Table II). We used Git as a remote repository to store the documents so that the dispersed team members can access them easily. When the main branch such as master is updated Git does not notify the user that the new/updated content is added but our system will alert the user as shown in Fig. 2 when the master branch got updated by any team member and the user can pull the changes in by simply clicking the notification bar. Sometimes the user creates documents in another directory which is not tracking by Git such as he/she writes a document in some folder and forgets to add this on project repository our system will identify the document which is modified in the system and will match with the documents of project documentation repository a notification will be generated shown in Fig. 3 that this document is related to that specific project click on it to copy that document in the specified project in the remote repository. Team members can share the documents via our tool.

For tagging mechanism, the system had a pre-loaded list of tags added by team leads during sprint meeting. Users can initiate the tag using # pre-loaded list of tags will be displayed then the user can select the relevant tag and continue chatting this conversation will be stored. Tags needs to be used only when the user thinks that the discussion needs to be saved that may be needed later via search feature we can search the tags,

auto complete list will be displayed to the user in the search bar user can select the tag and search against that tag and the related conversation will be displayed to the user including sender name, date time and the conversation main interface is shown in Fig. 4.

C. Case Study Results and Discussion

Our case study findings answers RQ3, we evaluated the interviews data and coded the data using Nvivo12 tool. We first created a mind map for our themes as shown in Fig. 5 and then we identified several themes in our data based on coding. The identified themes are "documentation problem", "knowledge vaporization issue", "reduced knowledge vaporization problem", "improved documentation", "improved knowledge sharing", "framework correctness". This theme helped us to understand what knowledge sharing problems they had and how they deal with it and what impact does our framework had on the company's knowledge sharing process Thematic analysis response is shown in Fig. 6.

By analyzing the data we had identified current knowledge sharing techniques in the company the team members use IM tools such as Slack and Email for communicating and sharing of knowledge, for video conferencing they use skype. The company faced documentation problem before presenting our framework our analysis revealed they face problems in documentation such as missing, poor and outdated documents. "We normally come across situations like file or documents placed somewhere in directories but we forget where we put them in the first place. Sometimes we face document inconsistency and redundancy because of no document management tool" the team leads described "There is a document management system missing in our company. As the company is ISO certified so detailed documentation is mandatory but an automated document management system is missing because of which we only manage documents manually, send them by emails and the manager store them in repository manually." They had outdated documents because there was no mechanism to update the document periodically the developers explained "we have to review old and new docs to maintain consistency in documents and we have to cross check our documents multiple times which is time taking also we had to maintain directories and need to make sure manually that everything is up to date and it wastes a lot of our energy." The company once used the Wiki as a repository but it did not resolve their issue of outdated documents. Manager told us "there are some outdated documents which we do not update once they are made. They just stay in the repository Still, the documents go missing." one of their developers told us "We use to face the problems like missing outdated and redundant documents (in few cases). Sometimes, it's hard to find the related document, as one couldn't recall the document name exactly and every document has some issues which need to be addressed. I review it several times."

Knowledge vaporization also existed in the company because they relied heavily on IM tools and sending/receiving emails there is a high tendency of misplacing a relevant document and when they need to recall some knowledge they need to manually search the old conversation which is a time-consuming process couple of developers told us "yes this is a huge problem for us as there is no mechanism to retrieve knowledge that is once shared in chats. we have to share it again on chat messengers, as in the free version of slack chats gets deleted when message limits to reachedto 10000, we either copy important points to notepad or we have to re-share them.", "We do face vaporization of knowledge I use to save the chats in notepad, we normally save important chats to notepad for future reference." Upon analyzing the data most of team members showed interest in some kind of mechanism to store and retrieve the knowledge to avoid knowledge vaporization problem. The team members of the company used our framework for two weeks and also they use scrum as an agile methodology their sprint consists of two weeks so they used our framework in their sprint session. The team members gives feedback of our proposed framework regard to documentation problem they described "it helps in reducing the time to find any document that is already stored somewhere in my system's hard disk by its document matching mechanism and it helps in updating the documents and it keeps track of every document and suggest the matching documents to save us from reinventing the wheel It seems useful if it is strictly followed by company. Otherwise, it will never give them desired results.", "it helps to reduce the redundancy of documentation by introducing documents type documents are always well managed and we can search them easily afterwards keeping documents randomly made them messy", "managing documents in categories make them more organized these document types are most common document types which are needed most of the times. So it's worthy to use". For knowledge vaporization, the team member's gives feedback after using the framework with a provided tool they described "it become easy to search the relevant knowledge using tags. shared knowledge can be retrieved from the tags which is very helpful for us", they said "somehow it solves the problem, but there will be some issues for non-technical person I think", "tags search mechanism is good to find desired shared knowledge it pretty much solves the problem by offering quick retrieval of knowledge by introducing tagging mechanism by using it we don't have to save chats in notepad", "It has allowed us to focus on development related problems. The tagging mechanism is a good facility. It's just like tagging on Facebook which make it easy to see only the tagged comment from millions of comments.", "I think so. Knowledge shared by my seniors/supervisor when stored in the form of tags make them more understandable and categorized, and to the extent that you get everything written down instead of verbal or informal communication." We asked the team members about any improvements they suggest so we can enhance the solution we asked about proposed framework number of activates what they think about it and what would they suggest if something did redundantly or missing in the framework they described "The proposed solution seems perfect when used among technical people by technical it means IT professionals., its document matching mechanism and tags search have solved many of our problems and it has increased our project development pace", "I think so knowledge vaporization is an important problem especially for those team who needs to work with other team members located at a remote location." The framework provides a centralized solution solving many of the small problems. we asked them if anything is missing?

They described "it seems good with current features. But can be improved for version controls (Git tracks the commits of users and it can be used further).", "This framework can be improved by adding individual chat along with group chats and I think we should be able to create different teams for different projects in chats. Although searching can be made stronger by providing more and more options that would be an extra feature and not a missing thing.", "I think Information Architecture (IA) and Product Description (PD) will overlap at some point and the features provided are good and not redundant. Each one of them is important in their own.", The team lead identified "if someone keeps changing a file in GIT repository, everyone will be pissed off from notifications with too much complex scenarios and iterations, I think this will lead to some redundancy on its own. Following improvements would be plus e.g. Adding more Document types, dealing with larger documents involving multiple tags search shared knowledge efficiently, Integration with other applications e.g. servers containing files and wikis There is always a place for further improvement the framework provides good options under one roof." In terms of usability of our tool blueprint, we obtained a SUS score of 83.6 (std =13.8), suggesting a high usability perception on the knowledge sharing tool. However, Fig. 7 presents this result using a curved grading scale [27], where we observed that around 9% of the participants perceived the knowledge sharing tool as having low usability (C and D grades on Fig. 7). The SUS [28] result suggests that participants perceived the knowledge sharing tool as highly usable. Which is further strengthn its importance for knowledge management as well [31-33] as explained by the authors.

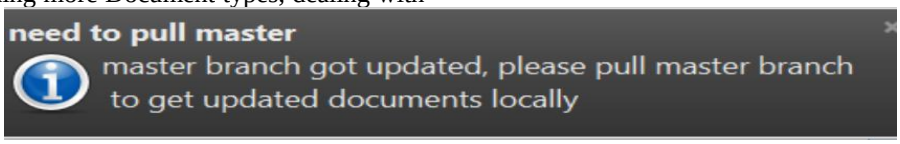

Fig. 2. Master Branch update Notification.

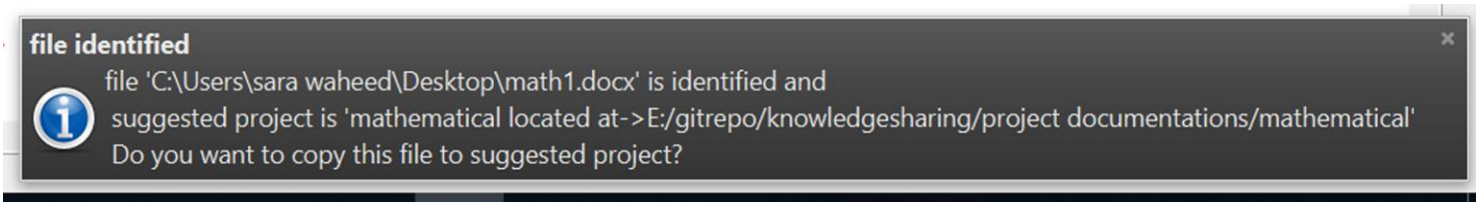

Fig. 3. File Identification Notification.

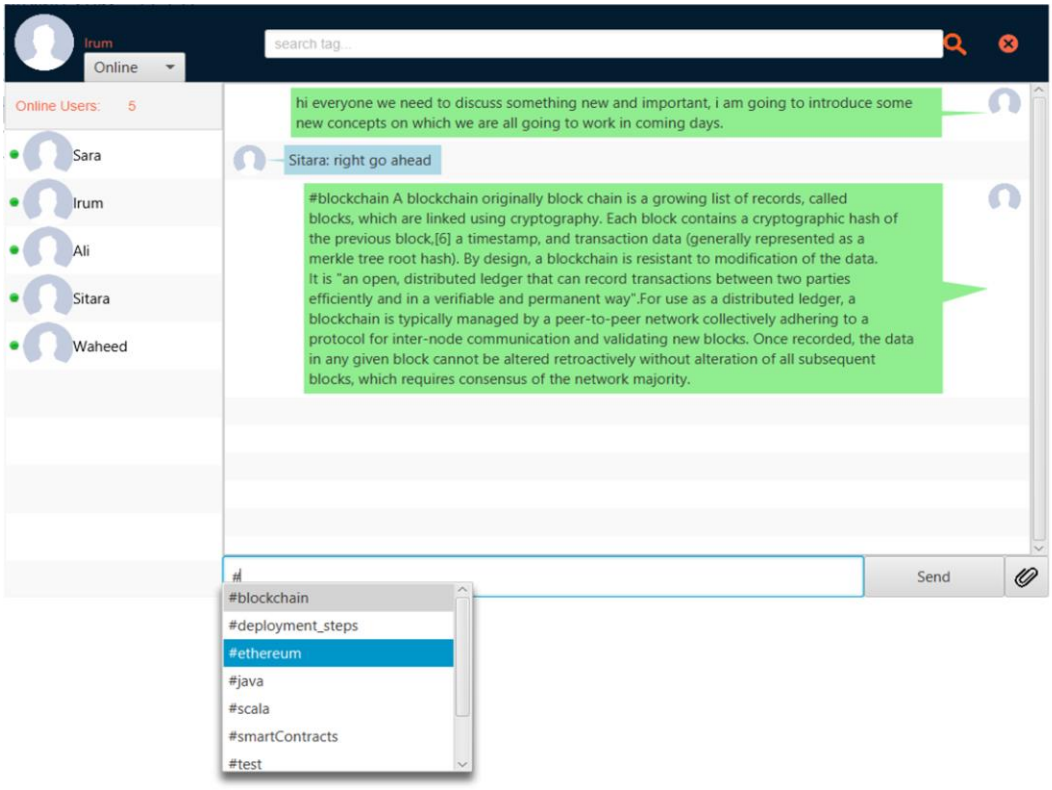

Fig. 4. Tool Support.

Fig. 5. Mind Map.

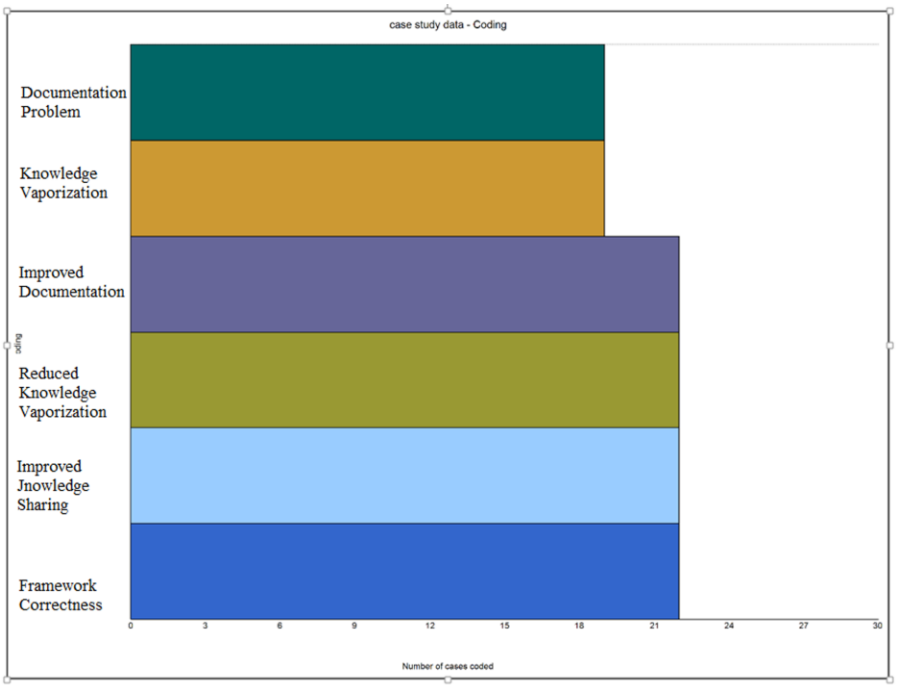

Fig. 6. Interview Resoponses.

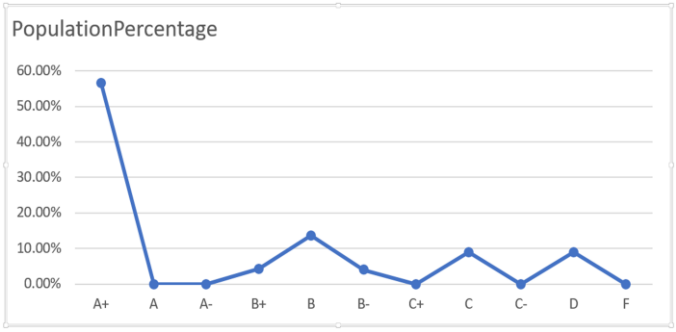

Fig. 7. SUS Result.

V. Limitation and Future Work

The proposed solution is evaluated via case study and it showed the positive results in improving knowledge sharing process but we evaluated our solution in a single context results may differ in other context because our solution is for distributed eniornments and as a future work there is always a possibility to improve the current solution to make it more suitable for distributed teams and they can share knowledge easily multi tags can be added for advanced search, documents can be tagged to make it more organized and searchable.

VI. Conclusion

In this research work, we focused on knowledge sharing which is an important phase of knowledge management the knowledge sharing process is itself complex in nature and it involves people, Knowledge sharing process becomes more critical when the development teams are dispersed around the globe. We identified several problems in knowledge sharing process when the teams are distributed geographically among all the challenges we identified documentation problem such as missing, poor and outdated documents and knowledge vaporization problem as much of the conversation and

communication is done via chat and emails and retrieving them later is a great headache. To eliminate these challenges we proposed a framework which deals with documentation problem and knowledge vaporization. We evaluated our framework by case study to evaluate frameworks performance in the real-life context, where actually the problem arises we conducted the interviews before and after describing our framework and came to the conclusion on team members response that they are satisfied with our proposed solution and it improved their knowledge sharing process. They indicated documentation improvement by having well managed, updated and complete documents and they also indicated reduced knowledge vaporization problem they used the tags in their daily conversation and retrieve these tags when required it produces less vaporization of knowledge. All these improvements positively increased knowledge sharing process which leads to fast product development and information flow in global software projects.

REFERENCES

[1] L. H. `Wong and R. M. Davison, "Knowledge sharing in a global logistics provider: An action research project," Information & Management, vol. 55, no. 5, pp. 547-557, 2018.

[2] B. Afsar, A. Shahjehan, S. I. Shah, and A. Wajid, "The mediating role of transformational leadership in the relationship between cultural intelligence and employee voice behavior: A case of hotel employees," International Journal of Intercultural Relations, vol. 69, pp. 66-75, 2019.

[3] D. Hislop, R. Bosua, and R. Helms, Knowledge management in organizations: A critical introduction. Oxford University Press, 2018.

[4] I. Rus, M. Lindvall, and S. Sinha, "Knowledge management in software engineering," IEEE software, vol. 19, no. 3, pp. 26–38, 2002

[5] M. Zahedi and M. A. Babar, "Knowledge sharing for common understanding of technical specifications through artifactual culture," in Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2014, p. 11.

[6] R. Anwar, M. Rehman, K. S. Wang, and M. A. Hashmani, "Systematic Literature Review of Knowledge Sharing Barriers and Facilitators in Global Software Development Organizations Using Concept Maps," IEEE Access, 2019.

[7] G. Borrego, A. L. Moran, and R. Palacio, "Preliminary evaluation of a ´tag-based knowledge condensation tool in agile and distributed teams," in Global Software Engineering (ICGSE), 2017 IEEE 12th International Conference on. IEEE, 2017, pp. 51–55.

[8] G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno, and F. O. García, "Towards a reduction in architectural knowledge vaporization during agile global software development," Information and Software Technology, 2019

[9] M. Zahedi, M. Shahin, and M. A. Babar, "A systematic review of knowledge sharing challenges and practices in global software development," International Journal of Information Management, vol. 36,no.6,pp. 995–1019, 2016.

[10] G. Borrego, "Condensing architectural knowledge from unstructured textual media in agile gsd teams," in Global Software Engineering Workshops (ICGSEW), 2016 IEEE 11th International Conference on. IEEE, 2016, pp. 69–72.

[11] H. Saint-Onge, "Tacit knowledge the key to the strategic alignment of intellectual capital," Planning Review, vol. 24, no. 2, pp. 10–16, 1996.

[12] E. D. Darr and T. R. Kurtzberg, "An investigation of partner similarity dimensions on knowledge transfer," Organizational behavior and human decision processes, vol. 82, no. 1, pp. 28–44, 2000.

[13] C. Wei Choo and R. Correa Drummond de Alvarenga Neto, "Beyond the ba: managing enabling contexts in knowledge organizations," Journal of Knowledge Management, vol. 14, no. 4, pp. 592–610, 2010.

[14] V. Santos, A. Goldman, and C. R. De Souza, "Fostering effective inter-team knowledge sharing in agile software development," Empirical Software Engineering, vol. 20, no. 4, pp. 1006–1051, 2015.

[15] C. Ebert and J. De Man, "Effectively utilizing project, product and process knowledge," Information and Software Technology, vol. 50, no. 6, pp. 579–594, 2008.

[16] S. Sarker, D. Nicholson, and K. D. Joshi, "Knowledge transfer in virtual systems development teams: An exploratory study of four key enablers," IEEE transactions on professional communication, vol. 48, no. 2, pp. 201–218, 2005.

[17] K. Stapel and K. Schneider, "Managing knowledge on communication and information flow in global software projects," Expert Systems, vol. 31, no. 3, pp. 234–252, 2014.

[18] A. Boden, G. Avram, L. Bannon, and V. Wulf, "Knowledge sharing practices and the impact of cultural factors: reflections on two case studies of offshoring in sme," Journal of software: Evolution and Process, vol. 24, no. 2, pp. 139–152, 2012.

[19] M. Bugajska, "Piloting knowledge transfer in it/is outsourcing relationship towards sustainable knowledge transfer process: Learnings from swiss financial institution," AMCIS 2007 Proceedings, p. 177, 2007.

[20] A. L. Chua and S. L. Pan, "Knowledge transfer and organizational learning in is offshore sourcing," Omega, vol. 36, no. 2, pp. 267–281, 2008.

[21] J. W. Rottman and M. C. Lacity, "A us client's learning from outsourcing it work offshore," in Information Systems Outsourcing. Springer, 2009, pp. 443–469.

[22] A. Boden, B. Nett, and V. Wulf, "Operational and strategic learning in global software development," IEEE software, vol. 27, no. 6, pp. 58–65, 2010.

[23] J. Bosch, "Software architecture: The next step," in European Workshop on Software Architecture. Springer, 2004, pp. 194–199.

[24] B. Kristjansson, R. Helms, and S. Brinkkemper, "Integration by communication: Knowledge exchange in global outsourcing of product software development," Expert Systems, vol. 31, no. 3, pp. 267–281, 2014.

[25] A. Averbakh, E. Knauss, and O. Liskin, "An experience base with rights management for global software engineering," in Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies. ACM, 2011, p. 10.

[26] J. Loeliger and M. McCullough, Version Control with Git: Powerfultools and techniques for collaborative software development. " O'Reilly Media, Inc.", 2012.

[27] J. Sauro and J. R. Lewis, Quantifying the user experience: Practical statistics for user research. Morgan Kaufmann, 2016.

[28] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the s{}`ystem usability scale," Intl. Journal of Human–Computer Interaction,vol. 24, no. 6, pp. 574–594, 2008.

[29] D. Damian and D. Moitra, "Guest editors' introduction: Global software development: How far have we come?" IEEE software, vol. 23, no. 5, pp. 17–19, 2006.

[30] R. F. Rich, "Knowledge creation, diffusion, and utilization: Perspectives of the founding editor of knowledge," Knowledge, vol. 12, no. 3, pp 319–337, 1991.

[31] Abbas, Syed Fakhar, Raja Khaim Shahzad, Mamoona Humayun, NZ. Jhanjhi, and Malak Alamri. "SOA Issues and their Solutions through Knowledge Based Techniques-A Review." International Journal of Computer Science And Network Security vol. 19, no. 1, pp.8-21, 2019.

[32] M Humayun, NZ Jhanjhi, "Exploring The Relationship Between Gsd, Knowledge Management, Trust And Collaboration", Journal of Engineering Science and Technology, vol. 14, issue 2,pp 820-843, 2019.

[33] S. S. A. Bukhari, M. Humayun, S. A. A. Shah and N. Jhanjhi, "Improving Requirement Engineering Process for Web Application Development," 2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), Karachi, Pakistan, 2018, pp. 1-5.