Knowledge Temple: A Collaborative Knowledge Sharing Technique for Agile Software Development

I. Burak Ersoy *Burak.Ersoy@tamucc.edu*
Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, 78412, USA

Ahmed M. Mahdy *Ahmed.Mahdy@tamucc.edu*
Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, 78412, USA

Abstract

In today's economy, enterprises require knowledge more than ever before. Employees are being classified based on their skill set and experience, where the tacit knowledge of individuals is a key factor. The effect of knowledge hunger can be easily seen in agile software development teams. To sustain the quality permanence of software development, it is essential to transform individuals' tacit knowledge to core organizational knowledge. To achieve this goal, every software development process utilizes different knowledge sharing and creation approaches. In this paper, a proposed technique, Knowledge Temple, is introduced as a feasible improvement to well-known knowledge sharing challenges for small agile software development teams. It is a hybrid technique, incorporating knowledge sharing and building models, such as cognitive apprenticeship, on-the-job-training, solo programming, pair programming, parallel peer programming, pair rotation, and knowledge repository creation. The proposed technique has been evaluated in the Innovation in Computing Research (iCORE) at Texas A&M University-Corpus Christi. Experimental results show that this hierarchical approach provides an iterative and incremental solution to sharing and creating knowledge in a collaborative and cooperative fashion.

Keywords: Software Engineering, Agile Software Development, Knowledge Sharing, Knowledge Creation, Knowledge Loss.

1. INTRODUCTION

Creating successful projects is the ultimate goal of software engineering. Thus, software development methodologies are introduced to overcome software development issues, such as late projects, budget issues, and faults [1]. Traditional software development methodologies, team software process (TSP) and personal software process (PSP) from the Software Engineering Institute (SEI) [2], and Agile methodologies [3] are leading software life-cycle models. Every life-cycle model offers different participation or learning activities, such as cognitive apprenticeship and knowledge repository creation routines. All those methodologies evolve around knowledge management; in fact, knowledge sharing is the major component of each.

Tacit knowledge is the experience of development, training, and/or education, which materializes in a person [4-9]. Software development is based on the tacit knowledge of the individuals. To sustain the quality permanence of software development, it is essential to transform individuals' tacit knowledge to core organizational knowledge. To achieve this goal, every software development process utilizes different knowledge sharing and creation approaches.

The problems of knowledge sharing and creation approaches are:

- Knowledge loss via retirement or high turnover rates and
- Knowledge hoarding for interpersonal reasons or organizational climate.

If the organization suffers from knowledge loss and knowledge hoarding, it may mean the organization is staff-dependent [10]. For organizational success and continuity, organizations have to be staff-independent. Being staff-independent means both knowledge loss and knowledge hoarding protected. In order to be staff-independent, organizations should share the knowledge among the development team. In addition, finding good programmers and the pace of technology change can be listed as knowledge sharing challenges [10].

The effect of knowledge hunger can be easily seen in agile software development teams. Biawowen [11] claims that we are in the "knowledge economy era" and states the knowledge necessity for agile software development teams in three steps:

1. Knowledge is the only meaningful resource,
2. Companies' products and services are based on the transformation of the knowledge, and
3. Software employees require more knowledge management than any other business sectors.

However, implementing knowledge sharing is not an easy task for agile development teams compared to its increasing demand. Therefore, surveying knowledge sharing issues through sociological, documentation, and implementation perspectives is essential to reveal the real motive [10].

Agile practices offer state-of-art solutions for knowledge building and sharing; however, they have their own drawbacks. In this paper, a proposed technique, Knowledge Temple, is introduced as a feasible improvement to well-known knowledge sharing challenges for small agile software development teams. It is a hybrid technique, incorporating knowledge sharing and building models, such as cognitive apprenticeship, on-the-job-training, solo programming, pair programming, parallel peer programming, pair rotation, and knowledge repository creation. This hierarchical approach provides an iterative and incremental solution to share and create knowledge in a collaborative and cooperative fashion.

2. KNOWLEDGE TEMPLE OVERVIEW

The paradigm shift from knowledge 'management' to knowledge 'sharing' has allowed software development teams to focus on the team members and their culture as much as their productivity. Maintaining productivity requires sustaining team member motivation, especially, for agile development teams. In addition, a good organizational culture transforms team development motivation to a successful knowledge sharing environment.

2.1 Evolution Knowledge Temple Practice

In order to create a knowledge sharing culture, pair programming was implemented with a small agile development team at the Innovation in Computing Research (iCORE). Three different types of progress were observed in every iteration cycle:

- Beginning of the iteration: low productivity and high knowledge sharing
- Middle of the iteration: medium productivity and low knowledge sharing
- Near the end of the iteration: high productivity and very low knowledge sharing

There was an inverse relationship between production level and knowledge sharing level. In every sprint, because of tight project deadlines and high turnover rate, high productivity and at least medium knowledge sharing was required. This requirement increased the responsibility burden of the expert developers. Both application development and knowledge exchange were fulfilled by the agency of expert developers, and it was the cause of their responsibility burden. To accomplish high levels of knowledge sharing, expert developers were paired with novice developers. It was the only way of growing the agile team because of the iCORE's developer resources.

The outcome of applying pair programming was not successful. It was either inadequate productivity and good knowledge exchange or good productivity and inadequate knowledge exchange. The novice programmers made lots of complaints about expert developers' availability. On the contrary, expert developers reported novice developers' motivation level as 'ground-level intentness.'

Even if pair programming was not the optimum choice, a natural apprenticeship instance between expert and novice developers occurred. The cognitive apprenticeship theory proceeded through expert mentoring rather than pair programming. However, it was not enough for carrying out the novice developers' contribution and sharing.

A middle-man was utilized between the expert and novice developers, creating the Knowledge Temple. The middle-man should:

- Free the expert developer to increase the productivity,
- Support the novice developer to stimulate learning curve,
- Contribute toward the development progress, and
- Hold up the development and knowledge sharing structure.

Consequently, small teams of three were formed and named as 'Temple.' Every Temple had a mandatory expert developer and two apprentices, entitled Temple Master and Temple Apprentices.

2.2 Knowledge Temple Technique

Having two apprentices under the influence of a lead created a core team culture. Cognitive apprenticeship theory is the dominant characteristic of the Knowledge Temple, as it is in human nature. The leadership of the Temple Master is as important as the will and autonomy of Temple Apprentices. However, the Temple Master has a high responsibility to sustain the Knowledge Temple mechanism. As shown in Figure 1, the Knowledge Temple contains three different zones addressing development and knowledge sharing.

Zone 1 is the Temple phase where the Temple Master and Apprentices perform solo programming. Zone 1 is extremely important for productivity when there is tight deadlines. The Temple Master should reserve development time, particularly when the development contribution of the Temple Apprentices is low. Moreover, project management meetings can be performed between the Temple Master and project manager as a zone 1 activity. While the Temple Master is in zone 1, the Temple Apprentices can stay in the phase of zone 1 or they can call for zone 2 between them. Being in the phase of zone 1 for the Temple Apprentices is essential both for development and knowledge building. The Temple Master has the privilege to assign duties, which can be a contribution for productivity, hands-on learning tasks, or a knowledge repository creation.

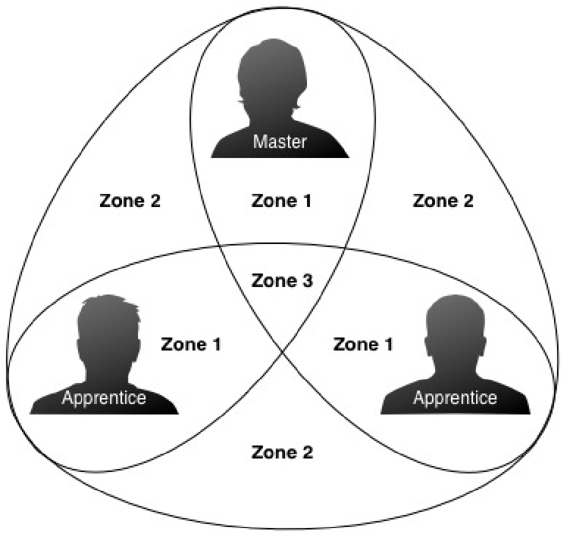

FIGURE 1: Knowledge Temple Technique.

Zone 2 is the phase where the Temple works as pairs. There are two ways of pairing: Master - Apprentice or Apprentice - Apprentice. The Master - Apprentice pairing allows higher productivity than knowledge sharing. On the other hand, the Apprentice - Apprentice pairing enables knowledge sharing more than productivity. In zone 2, pairs can perform on-the-job-training, pair programming, parallel peer programming, and knowledge repository creation. In addition, the nature of the Knowledge Temple technique facilitates pair rotation. Pair rotation can be performed in the Temple and among the Temples because an apprentice for Temple 1 can be an apprentice for Temple 2. For this reason, knowledge spreads in the agile development team like a social network.

In zone 3, both Temple Master and Apprentices come together and carry out activities as a team. Zone 3 is the core of the Knowledge Temple technique. It is the phase that lowers Temple member production, but highly increases the knowledge sharing and team building activities. The Temple Master creates the meeting agenda for the zone 3 phase through the progress of development. Temple members may engage in brainstorming, on-the-job-training, formal training, code revision, code inspection, Q&A sessions, or enhancing the communication between Temple members. The Temple forms a team structure in zone 3 to overcome the sociological issues of knowledge sharing. Furthermore, the project managers can be a part of the zone 3 meetings in order to monitor the Temple efficiency.

2.3 Building the Temple

The Temple initiation is an essential period in the life of the Knowledge Temple. Assigning the Temple Master among the agile development team is a simple but not easy task. It is simple because the selection process is related to the project and required development talents. Therefore, the number of available Temple Masters decreases through their required development experience. It is not easy because the Temple Master should have leadership and tutoring abilities to enhance knowledge sharing and team management. However, the team environment of Temples helps the Temple Master for both managing the team and maintain the development quality. After deciding the Temple Master, it is time to select the apprentices. The apprentice selection depends on the project requirements, which may demand:

- High productivity,
- A balance between productivity and knowledge sharing, or
- High knowledge sharing.

However, it is essential to keep the knowledge sharing level no less than medium because the high turnover rate is a concerning issue for all small agile development teams. Furthermore, a master may serve as an apprentice depending on the project requirements and expert skills.

The Temple, containing three expert team members, empowers high productivity. In this setting, the Temple Apprentices take more responsibility for application development. At the same time, they obtain more information about the project, the status of the project, and the development method of the project. They adapt faster for both the development and knowledge sharing phases. In addition, using three expert team members is a good way of growing new Temple Masters.

To create a balance between productivity and knowledge sharing, the Temple should contain one expert, one average, and one novice level developers. This is the best setting for the Knowledge Temple technique because it completely fulfills the middle-man approach. The Temple apprentice, who has an average level of experience, supports the other Temple apprentice for both development and knowledge sharing needs. Moreover, an average apprentice contributes to application development much more than a novice apprentice, which makes The Temple Master's job easier. At the same time, he learns from the Temple Master quicker than a novice apprentice, and shares the knowledge with the novice apprentice efficiently.

An expert and two novice developers form the Temple for high knowledge sharing. Two Temple Apprentices, who have almost the same level of experience, enables a strong learning environment. Even if their contribution to development is minor compared to other Temple alternatives, the Temple Apprentices have a strong motivation to exchange knowledge. This ambitious impulse provides a big potential for future projects. In addition, two novice Temple Apprentices may affect the Temple Master's productivity. However, the Temple develops more innovative approaches due to an increased number of Temple meetings.

2.4 Knowledge Temple vs. Jedi Temple

The fun factor of agile development is also indispensable. It encourages team building and team unity. Star Wars^{TM} was selected as the theme of the Knowledge Temple technique (Figure 2).

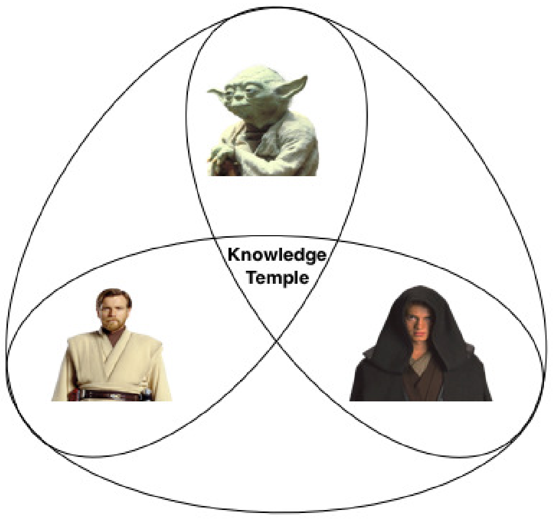

FIGURE 2: Knowledge Temple vs. Jedi Temple. Images are sourced from: http://www.starwars.com/.

There are two reasons behind the Star Wars^{TM} theme. First, Star Wars^{TM} is a very popular movie among people in science and technology [12-14]. Therefore, introducing a new technique with a well-known and beloved theme allows an immediate adaption. Second, the nature of the selected

Star Wars^{TM} characters are self-descriptive for both the Temple roles and the Knowledge Temple mechanism.

The Yoda character in the Star Wars^{TM} universe is selected as the Temple Master in the Knowledge Temple technique, due to his leadership, mentorship, and high talents. The Obi-Wan and the Anakin characters are the Temple Apprentices in the proposed technique because of their cooperative and collaborative efforts in the Star Wars^{TM} universe. Through the selected characters, the agile development team conceptualizes the roles and the role hierarchy in the Knowledge Temple technique. Moreover, the interaction between the selected Star WarsTM characters describes the the Knowledge Temple mechanism. Yoda, Obi-Wan, and Anakin may accomplish quests as a tightly-coupled team, loosely-coupled teams, or solo heroes. They have their individual and team responsibilities and report to each other through their hierarchy. They are always determined in their quests, eager to learn the power of the Force, and respectful to each other. As a result, the ambiance of the Jedi Temple in the Star WarsTM universe is the ideal scene for the Knowledge Temple technique.

3. EXPERIMENT DESCRIPTION

Software engineering is a developing practice compared to other engineering fields or science disciplines. Even if software engineering is still an immature regimen, it has progressed very far in a short amount of time along with new software engineering branches. Agile software engineering is one of the most challenging and promising areas for empirical software engineering research. The nature of agile methodologies requires informal, observational, and on-the-job research. Therefore, empirical studies offer an essential way to evaluate new agile approaches. However, researchers argue about the contributions of empirical software engineering research [15] and offer ground rules to improve the results of empirical studies [16,17].

In order to improve the research and reporting processes, the Empirical Research in Software Engineering Guideline designed by Kitchenham et al. [17] was followed. The researched characterization framework introduced by Shaw [15] and the empirical software engineering research best practices from Weyuker [16] were also considered and utilized. In addition, the Knowledge Survey, which was developed by Palmieri [18], was put into practice as a research and evaluation method.

3.1 Experiment Context

Pair programming is a successful knowledge sharing technique if its requirements are all fulfilled. For a small agile development team, however, applying pair programming causes utter confusion between productivity and knowledge sharing for the pairs. The tight schedule of application development does not allow lead contributors to share their tacit knowledge with newcomers. Moreover, knowledge hoarding issues increase if the small development team has a high turnover rate. As a result, the small agile development teams may not create harmony for a collaborative and cooperative working environment through pair programming.

In this work, the possibility of a new knowledge sharing technique is discussed, considering the well-known pair programming issues. To enable a collaborative production, the experience gap between the pairs was focused on. Augmenting the knowledge transfer potential was sought, while development productivity was ensured. The issue of scheduling was delved into through development deadline and knowledge sharing burden. Finally, the team determined to create a knowledge sharing culture, which constantly increases team motivation in an agile environment.

3.2 Experiment Population

The Knowledge Temple was applied in iCORE at Texas A&M University-Corpus Christi. iCORE is a research, development, and commercialization group, which comprises undergraduate and graduate level students. The agile development team of iCORE was formed from sophomore, junior, and senior level undergraduate and Master's level graduate students. The team did not include freshman level undergraduate students due to their insufficient programming abilities. All the students were part-time workers, who contributed ten or twenty weekly work hours as a part of the agile development team.

The varied levels of computer science students created an environment that could be considered as a real world atmosphere:

- Sophomore and junior level undergraduate students as newly-hired developers or interns,
- Senior level undergraduate students as junior developers, and
- Master's level graduate students as senior developers.

Therefore, iCORE offered a unique empirical research environment for an observational experiment. Moreover, it is essential to have a diverse group of team members to effectively evaluate knowledge sharing results. It is assumed that the expert developers have more experience on project development requirements than apprentice developers in Knowledge Temple experiment. This unique environment exposed a mandatory employee turnover rate through the graduation of team members.

The cultural diversity of iCORE also offered an outstanding research environment. The experiment population contained team members from the United States, Vietnam, India, and Turkey. It allowed for the creation of a melting pot of different cultures and work ethics. In addition, applying the proposed technique in a university environment was promising because today's students will be tomorrow's professionals; thus, it was important to get results from future generations.

The agile development team had fifteen members. Each team member named as TMb# (Team Member #), where "#" stands for both the sequence of recruitment and ID number. For instance, TMb1 joined the development team first and TMb18 was last. The numbering system is important to comprehend the evolution of the team members. Nonetheless, it does not show the experience difference between team members because there is an opportunity that a Master's level graduate student can join the agile team after a sophomore level undergraduate student or vice versa.

3.3 Experiment Projects

The experiment environment had different levels of developers and different types and levels of projects. The proposed knowledge sharing technique was applied to three different projects. One of the projects was examined through version 0, version 1 and version 2 standings. Moreover, the proposed approach was applied not only to programming but to every aspect of the project development process, such as client collaboration, application publishing, and project presentation. SOAR SI, CCISD, and Museum were the names of the projects.

The SOAR SI project was an informative mobile application for science, technology, engineering, and mathematics (STEM) undergraduate students. It offers schedule, location, and orientation about supplemental instruction (SI) sessions offered by the Title V-STEM Outreach, Access, and Retention (SOAR) Program at Texas A&M University-Corpus Christi. The application contains six touch user interfaces (TUI) and nine development modules. The development team utilized the Appcelerator Platform and the Titanium SDK as the mobile application development platform. The SOAR SI project was published for both iOS (iPhone and iPad) and Android (smartphones and tablets) devices.

The CCISD project is a full educational guidance application for Corpus Christi Independent School District (CCISD). It presents a school directory, CCISD school calendar, CCISD lunch menu, CCISD news, CCISD athletics, reporting a bully functionality, and more. The application contains twelve TUIs and fourteen development modules. The development team utilized the Appcelerator Platform and the Titanium SDK as the mobile application development platform. The CCISD project was developed for both iOS (iPhone and iPad) and Android (smartphones and tablets) devices.

The Museum project is a full body interactive wall with custom design exhibits for the Corpus Christi Museum of Science and History. It introduces a dynamic projected content on the museum wall for children through interactive science and history education. Adobe Flash Professional and GroundFX Flash SDK from GestureTek were selected as the development platforms. The Museum project was under prototyping process, which was designed for a special interactive wall projection system.

3.4 Experiment Technologies

The use of technology is a driving force for software engineering methodologies. Especially for agile development, there is a skyrocketing market for different methods, conditions, and settings. The Knowledge Temple presents a knowledge sharing technique; however, building knowledge sharing culture within the organization and beneficial technology solutions for the agile development team are the beginnings of success. Therefore, any technological tool that works for the Temple was the point of interest. It was also important to build a balance for the flexible operating manner for the Temples.

In iCORE, it is a rule to use Bitbucket as a version control system. Bitbucket, a web-based hosting service for projects, allows public and private project repositories, team management, code reviews, and source code insight. Therefore, the Temple development and sharing progress was tracked by the developer submissions, assigned issues, Wiki, and comments through source code reviews. However, some Temples also took advantage of Trello for their project management purposes.

For mobile development, the Appcelerator Platform was used. The Titanium SDK employs only JavaScript language for creating native applications across different mobile devices. Using one development language for both iOS and Android development accelerated the development iterations. Moreover, the modular development design of Titanium allowed the team to build an on-the-job knowledge sharing culture through code modules. Another script-based platform, Adobe Flash Professional is also used in this experiment.

For documentation purposes, the development team suggested using the JSDoc documentation tool. It is an inline API documentation tool for JavaScript. Therefore, the Temple members added documentation comments to source code to create Wikis for knowledge sharing fashion.

To enhance communication and collaboration, the development team facilitated different video conferencing tools. They made use of Skype and Google Hangouts occasionally. However, TeamViewer was the widely used tool to establish a flexible time-sharing. The screen sharing and browser-based presentation features were indispensable and formed a robust learning environment.

In addition to software products, the team employed Alienware 23-Inch Desktops, 21-Inch iMacs, a projector, and a multi-touch smart board. The desktop computers were put in practice for development, testing, and knowledge sharing. The projector was used when Temples met in the brainstorming area at iCORE. Nonetheless, the most engaging learning tool was the smart board, in the iCORE conference room, because team members performed knowledge sharing, testing, informative Temple meetings, and customer collaboration with the help of the smart board. It increased the team motivation and empowered application interaction with its multi-touch feature.

3.5 Experiment Questionnaire

In addition to observational research, a survey was utilized as one of the research methods because of the high developer turnover in iCORE. Most of the participants had graduated and started to work in different parts of the United States while the progress of the experiment was being observed.

Palmieri [16] developed the Knowledge Survey to assess the experiment of using pair programming as a knowledge management technique. It was essential to use a survey that had already proved reliable and valid. All the questions were closed-ended to offer the same mental set while answering the questionnaire. The questions were kept as similar as possible to perform a similar questionnaire concept to the proposed solution. The survey was divided into 3 sections:

- Section 1: Knowledge Sources
- Section 2: Knowledge Acquisition, Dissemination, and Maintenance
- Section 3: Demographical Background Information

In Section 1, the questions were designed to investigate the sources utilized instead of emphasizing knowledge sharing terminology. In addition, Section 1 questioned the tools and knowledge sharing procedures that the proposed technique should evaluate. Section 2 investigated the organizational strategy on knowledge sharing. Additionally, Section 2 inquired about the effect of the proposed technique on knowledge hoarding and employee turnover. Finally, Section 3 had questions to capture the demographical background information of the team members.

Required sections were modified in order to fulfill the experiment context. In Section 1 and Section 2, some questions were deleted due to the Knowledge Temple technique progress and experiment resources. In Section 3, the question that directly related with pair programming was deleted and two questions were added instead:

- How satisfied are you working in a small team with 2 peers?
- How satisfied are you working in a small team with a peer?

The newly added questions investigated the receptiveness of team members, who worked in a small group of three people, to each other and the method. Participants were asked to reply through their satisfaction level. Their feedback formed the fundamental results of our research and the foundational theory for future studies.

4. EXPERIMENT RESULTS

Evaluating empirical software engineering research was a complicated process. To acquire reliable results, the contribution of Temple members was analyzed through the Bitbucket platform and a questionnaire was administered after the Knowledge Temple experiment at iCORE. Moreover, observational experiences from applying the Knowledge Temple technique in a small agile development team was shared. The Knowledge Temple questionnaire was a variation of a questionnaire by Palmieri [18], which was a main instrument used to evaluate the effect of pair programming as a knowledge management approach. Some questions were altered to best fit the proposed Knowledge Temple technique [19]. Moreover, preliminary analysis was performed on the data to ensure integrity.

4.1 Team Member Contribution

Bitbucket is a web-based hosting service, which offers revision control, code insight and code review. Therefore, Bitbucket was utilized for development, knowledge sharing, and evaluating the development progress. All the team members had their individual accounts on the Bitbucket system. Through this account, they accessed the code of the project and contributed to the project by module development. Whenever a team member accomplished a working copy of the module, s/he submitted this version through the version control system. As shown in Table 1, the number of submissions from Temple Masters and Temple Apprentices was evaluated. In addition, the production modules were designed with as comparable size and complexity as possible. This was largely feasible due to the nature and type of the projects that the experiment was conducted on.

For SOAR SI version 1, 256 submissions were recorded in Bitbucket. The three Temple Masters completed 141 submissions and the eight Temple Apprentices achieved 115 submissions. As a result, the Temple Masters developed 55% of SOAR SI version 1. It was expected that the three Temple Masters lead productivity; however, the contribution from the eight Temple Apprentices was higher than expected. The reason for this was the on-the-job learning culture of the Knowledge Temple technique.

Project Name	Total Submissions	Temple Masters	Temple Apprentices
SOAR SI Version 1	256	141 (55%)	115 (45%)
CCISD	115	65 (56.5%)	50 (43.5%)

TABLE 1: Submission Results.

In the CCISD Project, 115 submissions have been archived at the time of this research. The CCISD project is still under development. The Temple Master completed 65 submissions and the two Temple Apprentices achieved 50 submissions. Consequently, the productivity contribution of the Temple Master was higher (56.5%) than the two Temple Apprentices (43.5%) as expected. However, this result established the fact that the Temple was exchanging and building knowledge through the Knowledge Temple technique process while they were developing software. Moreover, the Temple Master acknowledged that the Temple Apprentices were able to apply the gained knowledge from the former project into the CCISD project.

4.2 Questionnaire Results

Fifteen responses for the Knowledge Temple Questionnaire were received, which was the total number of experiment participants. Through the single-blind experiment approach, the questions could not be asked directly regarding the Knowledge Temple technique. The first section of the survey investigated the participants' knowledge source penetration for knowledge building behavior (Figure 3). Internal publications, design documents, external publications, the Internet, co-workers, classroom or online courses, and databases or groupwares as a knowledge source were the foci. The participants indicated almost equal usage of internal publications (77%), design documents (73%), external publications (66%), and classroom or online courses (74%) for knowledge sharing and knowledge building purposes. However, the Internet (100%), databases or groupwares (87%), and co-workers (84%) were voted as the most widely used sources. Therefore, utilizing the Internet and web-based knowledge sharing tools, such as Bitbucket, Dropbox, Google Drive, TeamViewer, Skype, or Google Hangouts, was a very important research perspective for the Knowledge Temple experiment. Moreover, combining the power of the web with the competence of the participants' co-workers created an influential knowledge sharing culture. The participants also recommended workshops and hands-on studies for the open-ended other beneficial knowledge sources question.

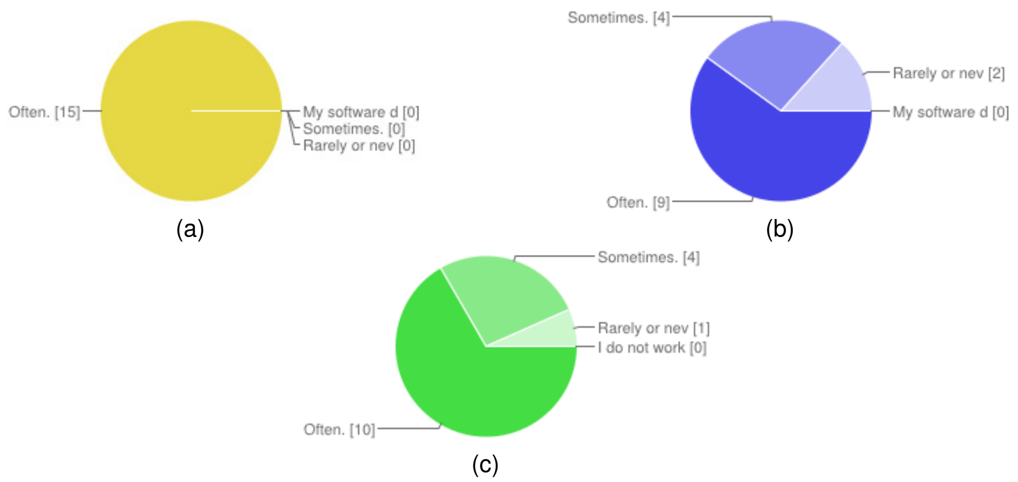

FIGURE 3: Knowledge Sharing Sources. (a) The Internet; (b) Databases and Groupwares; (c) Co-workers.

In Section 2, the survey targeted the participants' knowledge sharing determination and knowledge lost perspective through the knowledge acquisition, dissemination, and maintenance processes.

As shown in Figure 4, most of the participants (73%) thought iCORE's agile development team was good at creating new knowledge through its people and technical resources. However, they (47%) argued that the team was not adequate by means of finding, organizing, and documenting the knowledge possessed through the Knowledge Temple technique. Although 53% of the participants were satisfied with the knowledge creation process, the percentage was expected to be closer to 80%; therefore, this is an area that requires more research. The analysis supports both effective knowledge acquisition from outside sources and effective knowledge accessibility. Utilizing web-based repositories and web-based communication allowed a continually approachable environment.

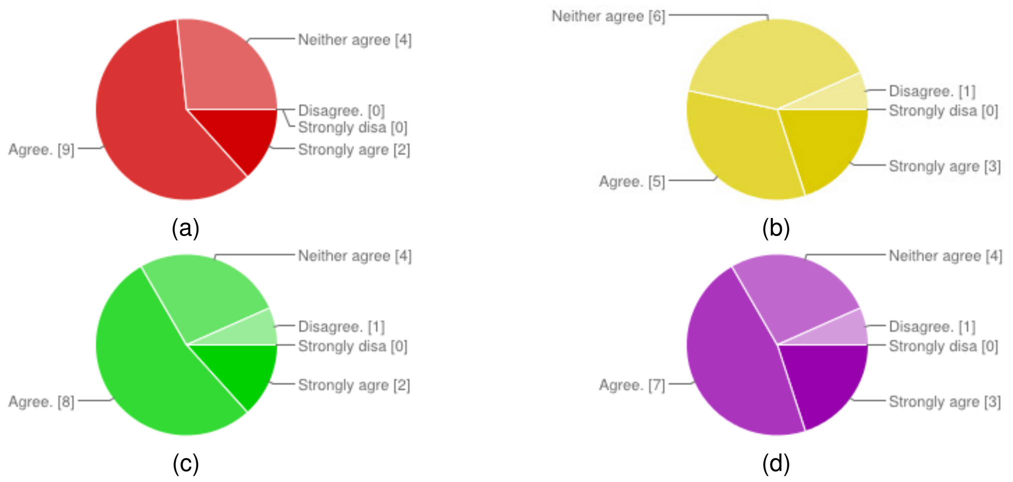

FIGURE 4: Knowledge Creation and Accessibility. (a) Creating Knowledge through People and Technological Resources; (b) Finding, Organizing, and Documenting the Knowledge; (c) Acquiring Knowledge from Outside Sources; (d) Making Knowledge Accessible for Anytime, Anywhere.

Testing the effects of the Knowledge Temple technique on knowledge hoarding problems was essential considering influential knowledge sharing. As shown in Figure 5, the impact of knowledge sharing among the iCORE team members was highly acknowledged. Analysis of development quality and productivity assessment question showed that 93% of the survey participants improved their development abilities due to knowledge sharing. Although participants (79%) stressed that their unique knowledge enhanced their competitive advantage over their peers, 66% of the participants gladly agreed to share knowledge with the agile development team. Moreover, the flexible productivity and knowledge sharing culture of the Knowledge Temple technique created an extraordinary workplace where teams and individuals worked simultaneously. The results of the question about the performance rewarding process showed that 50% of the participants determined team success and individual accomplishment are equally effective. Counterbalancing this, an equal amount of participants argued the case considering "primarily individual accomplishments, but also some team success" (21%) or "primarily team success, but also some individual accomplishments" (21%).

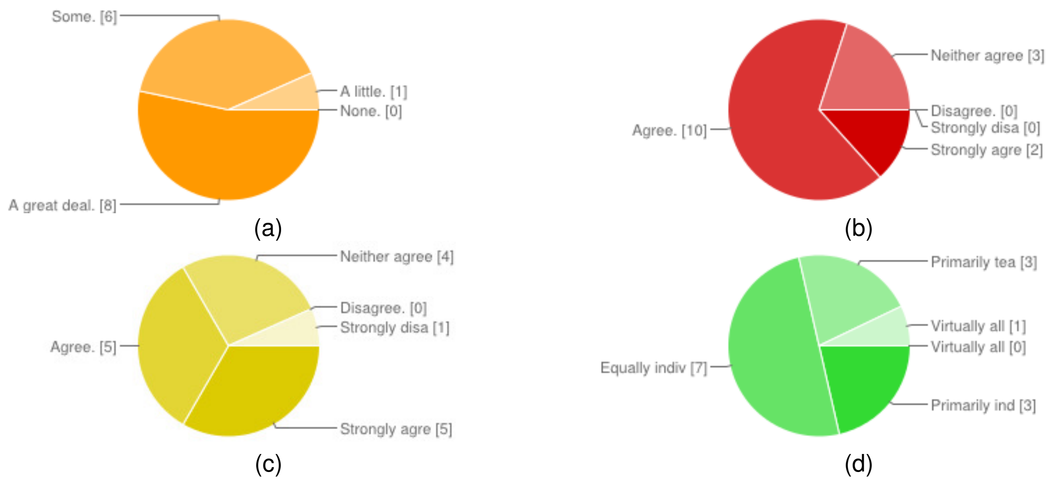

FIGURE 5: Knowledge Hoarding Effects. (a) Improving the Quality and Productivity due to Shared Knowledge; (b) Hoarding Knowledge because of Competitive Advantage Over Team Members; (c) Willing to Share the Knowledge Among the Team Members; (d) Rewarding based on Individual Technical Accomplishments vs. Team Success.

Experiencing mandatory employee turnover was a challenge for the Knowledge Temple experiment; however, the experiment participants presented courage and confidence through compensation of knowledge loss due to team member turnover (Figure 6). According to the results, team members can compensate for knowledge loss of less than a month (46%) or between one month and three months (33%). Moreover, the effect of losing an expert member did not change the trust of residual agile development team members. The team members (47%) admitted to covering the knowledge loss due to expert member turnover.

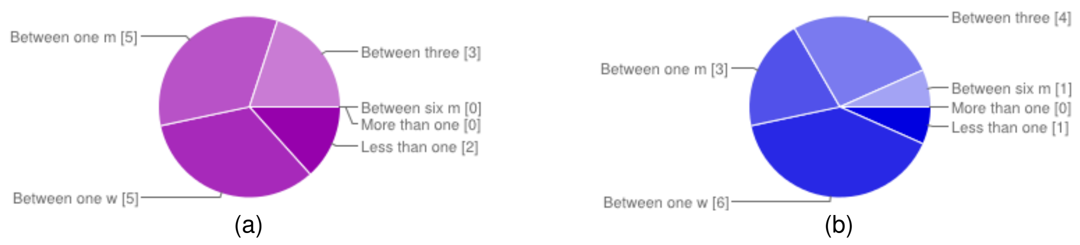

FIGURE 6: Knowledge Loss Effects. (a) Compensating the Knowledge Loss due to Team Member Turnover; (b) Compensating the Knowledge Loss due to Senior Technical Member Turnover.

In Section 3, the sociological factors in the workplace were evaluated. The participants (87%), who performed the Knowledge Temple technique, were satisfied with their working environment. As shown in Figure 7, 66% of the participants presented collaborative work between 10% and

50% of their daily working hours. Moreover, the agile development team members felt confident in a working environment both with a peer (93%) and with 2 peers (86%).

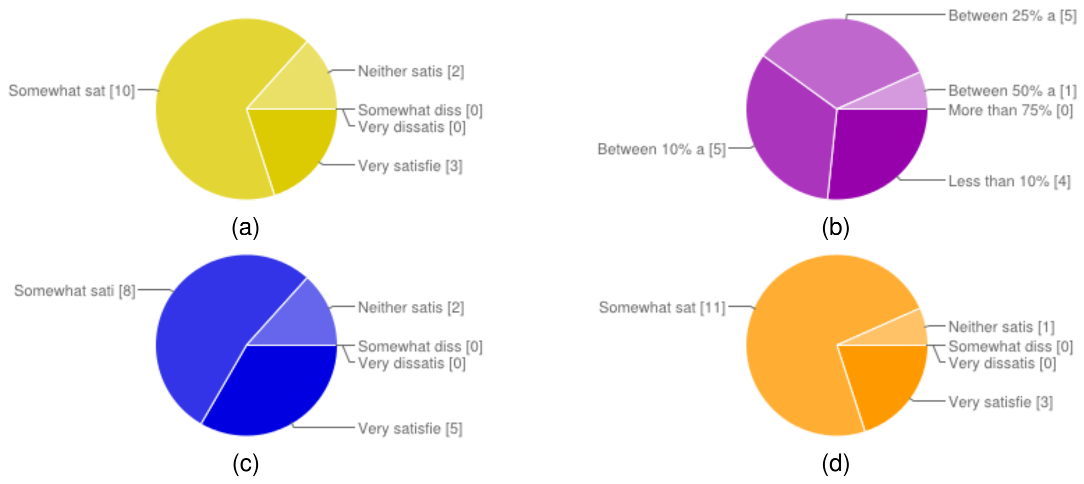

FIGURE 7: Demographical Workplace Information. (a) Enjoying the Current Position; (b) Collaborating with Team Members; (c) Working in a Small Team with 2 Peers; (d) Working in a Small Team with a Peer.

4.3 Observational Results

Experiencing the drawbacks of pair programming changed the development perspective. The programming level difference of the agile development team members did not allow for the application of pair programming successfully. However, the power of pair programming was not undervalued. The development team members required a flexible working environment where they could accomplish both application development and knowledge sharing. The iCORE environment empowered the Knowledge Temple technique with its nature. Having different levels of programmers facilitated a working environment as a team of three. Therefore, the expert developer could continue development and share knowledge. At the same time, the average and novice developers could build knowledge and contribute to application development. Having three developers in a team, influenced by the expert developer, allowed much more adaptable and responsive team work.

In the Knowledge Temple experiment, one of the most important decisions was selecting the Temple Master. The Temple Master was responsible for the Temple and controlled the Temple through guidance. Every Temple had its own rules and way of accomplishing requested job duties. However, the Temple Master was the one who ensured the productivity of the project and knowledge sharing progress among the team. This management allowed the Temple Apprentices to contribute more while they were learning through their own efforts, pair studies, or Temple unification.

The selected theme, Star Wars^{TM}, notably increased the motivation of the Temple Apprentices. The idea of being Yoda was a big impulse compared to being a leader or a master for a team. It is worth noting that Star Wars^{TM} may not always be the best theme for any development environment. Therefore, another theme could be selected if required. However, it is important to choose a theme that can conceptualize the hierarchy, the mechanism, and the communication of the Knowledge Temple technique.

The unique environment of iCORE offered a high employee turnover through graduation of the team members; therefore, it was difficult to observe the effects of the Knowledge Temple technique for employee turnover. However, the graduated members reported that they felt the loss of the team working environment at iCORE.

Another challenge in iCORE was the tight deadlines of the agile projects. Therefore, the Temple of three experts or the Temple of one expert, one average, and one novice were created for most of the projects. The Temple of three experts hastened the development speed of the projects and assisted Temple Master growth for different Temples. On the other hand, the Temple of one expert, one average, and one novice enhanced the productivity and knowledge sharing simultaneously. It was also the best fit for the varied levels of iCORE developers. The Temple of one expert and two novices was also utilized in the Knowledge Temple experiment. This Temple style enabled the knowledge sharing and active learning for the novice developers; however, productivity decrease was reported by Temple Masters. Therefore, building the Temple was an essential part of the Knowledge Temple technique requiring a good knowledge level observation among the development team members. Moreover, the project requirements influenced the Temple building process through appropriate developer selection.

The Knowledge Temple technique built a working culture for iCORE. The hybrid setting of the Knowledge Temple technique streamlined the agile development team members by the adaptation process. The amenity of the Knowledge Temple mechanism simplified the knowledge sharing process. Traditionally, newcomers hoarded the knowledge that they possessed through application development; however, the iCORE culture oriented all the team members to team success rather than individual accomplishments.

Finally, small agile development teams require internal growth from their developers in the areas of development continuity and quality, due to the difficulty in hiring external competent developers. The Knowledge Temple technique facilitates the team to share and build knowledge between team members. Having three different zones to communicate, contemplate, and develop escalates the growth process of successful developers for small agile development teams.

5. CONCLUSION AND FUTURE RESEARCH

Despite the productive, flexible, and adaptive nature of agile development, it may suffer from knowledge sharing limitations. This includes knowledge loss due to retirement or high turnover rates of skilled professionals and knowledge hoarding due to interpersonal or organizational climate. Thus, the internal growth of developers is highly desirable for a small development team to maintain production quality. The influence of pair programming for software development and knowledge sharing is respected. However, this technique is confronted by time-sharing issues, due to attempting to perform a number of tasks concurrently; motivational loss issues, due to pair level difference; and focus shift to separated tasks instead of a common goal, due to tight deadlines.

The Knowledge Temple was proposed as a knowledge sharing technique for small agile software development teams that supports both software development productivity and knowledge exchange between team members. The Knowledge Temple is a cognitive apprenticeship model, where every Temple has three members: one Temple Master and two Temple Apprentices. There are three zones where Temple members can perform software development and knowledge sharing methods, such as on-the-job-training, solo programming, pair programming, parallel peer programming, pair rotation, and knowledge repository creation.

A single-blind experiment was performed with iCORE at Texas A&M University-Corpus Christi. Almost all of the development team members were part-time working university students either undergraduate or graduate level. The Knowledge Temple technique was administered in three different projects with ten varied Temples. To evaluate this empirical thesis study, Temple member's development contributions, a Knowledge Temple questionnaire, and observational outcomes were utilized. The results of the Knowledge Temple experiment illustrated:

- Development priority for Temple Masters,
- Knowledge sharing availability for Temple Masters,
- Knowledge sharing priority for Temple Apprentices,

- Development support opportunity for Temple Apprentices,
- Flexible scheduling for both development and knowledge sharing processes,
- Inspirational small team fashion, and
- Motivation continuity through Temple member availability.

Consequently, team member contribution, questionnaire results, and observational results yielded significant evidence that the Knowledge Temple technique for small agile development teams is an effective means of software development and knowledge sharing simultaneously. Moreover, the iCORE team members noted that they enjoyed the experience and declared that their technical skills had been increased. However, this empirical study alone is insufficient to validate the reported benefits of this knowledge sharing and development style. The Knowledge Temple technique should be performed as a teaching technique in academia to evaluate the influence on future generations. In addition, as mentioned previously, a higher number of participants were expected to be satisfied with the knowledge creation process; however, only 53% were satisfied with this process. Therefore, this difference needs further exploration. Finally, examining the proposed technique in the industry with full-time workers is another way to comprehend the collaborative and cooperative effects of the Knowledge Temple technique.

Acknowledgments
This work has been supported, in part, by National Science Foundation grant CNS-1042341.

6. REFERENCES

1. Chau, T., Maurer, F., and Melnik, G. Knowledge sharing: agile methods vs. tayloristic methods. In Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on (2003), pp. 302–307.

2. Amaral, L., and Faria, J. A gap analysis methodology for the team software process. In Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the (2010), pp. 424–429.

3. Chowdhury, A., and Huda, M. Comparison between adaptive software development and feature driven development. In Computer Science and Network Technology (ICCSNT), 2011 International Conference on (2011), vol. 1, pp. 363–367.

4. Abdullah, R., and Talib, A. Knowledge management system model in enhancing knowledge facilitation of software process improvement for software house organization. In Information Retrieval Knowledge Management (CAMP), 2012 International Conference on (2012), pp. 60–63.

5. Crawford, B., Castro, C., and Monfroy, E. Knowledge management in different software development approaches. In Advances in Information Systems, T. Yakhno and E. Neuhold, Eds., vol. 4243 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 304–313.

6. Jiang, H., Liu, C., and Cui, Z. Research on knowledge management system in enterprise. In Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on (2009), pp. 1–4.

7. Salleh, K. Tacit knowledge and accountants: Knowledge sharing model. In Computer Engineering and Applications (ICCEA), 2010 Second International Conference on (2010), vol. 2, pp. 393–397.

8. Stettina, C., Heijstek, W., and Faegri, T. Documentation work in agile teams: The role of documentation formalism in achieving a sustainable practice. In Agile Conference (AGILE), 2012 (2012), pp. 31–40.

9. Tao, Y., Wang, J., Wang, X., He, D., and Yang, S. Knowledge-based flexible business process management. In TENCON 2006. 2006 IEEE Region 10 Conference (2006), pp. 1–3.

10. Ersoy, B., and Mahdy A. Agile Knowledge Sharing. International Journal of Software Engineering (IJSE) 2015, 6, 1-15, 1.

11. Biao-wen, L. The analysis of obstacles and solutions for software enterprises to implement knowledge management. In Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on (2010), pp. 211–214.

12. Briggs, J. "star wars", model making, and cultural critique: A case for film study in art classrooms. Art Education 62, 5 (2009), 39 – 45.

13. Kapell, M., and Lawrence, J. Finding the Force in the Star Wars Franchise: Fans, Merchandise, and Critics. Popular culture and everyday life. Peter Lang Pub Incorporated, 2006.

14. Roberts, A. Culture, identities and technology in the Star Wars films: Essays on the two trilogies. SCIENCE-FICTION STUDIES 35 (n.d.), 156 – 159.

15. Shaw, M. What makes good research in software engineering? In Presented at the European Joint Conference of Theory and Practice of Software (ETAPS 2002), Grenoble, France. To appear in the International Journal on Software Tools for Technology Transfer. (2002).

16. Weyuker, E. Empirical software engineering research - the good, the bad, the ugly. In Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on (2011), pp. 1–9.

17. Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., El Emam, K., and Rosenberg, J. Preliminary guidelines for empirical research in software engineering. Software Engineering, IEEE Transactions on 28, 8 (2002), 721–734.

18. Roberts, A. Culture, identities and technology in the Star Wars films: Essays on the two trilogies. SCIENCE-FICTION STUDIES 35 (n.d.), 156 – 159.

19. Palmieri, D. W. Knowledge Management Through Pair Programming. PhD thesis, North Carolina State University, 2200 Hillsborough, Raleigh, NC 27695, 2002.