2017 24th Asia-Pacific Software Engineering Conference

A Goal-Driven Framework in Support of Knowledge Management

Guoping Rong†, Xinbei Liu‡, Shenghui Gu§, Dong Shao†
Software Institute, Nanjing University, Nanjing, Jiangsu, P.R.China
†{ronggp, dongshao}@nju.edu.cn
‡141250078@smail.nju.edu.cn §SamuelGarciaSTK@gmail.com

***Abstract*—Knowledge management nowadays usually focuses on the choice among some models or methodologies as a whole, but not on some specific, quantitative contributions of particular goals of the organization. Such a simplification misses some important chances for knowledge integration and transformation. What's worse, this simplification depresses the motivation of team members to accumulate and use the knowledge. In this paper, we propose a knowledge management framework which features in its goal-driven philosophy to manage project development, organize the knowledge and effectively integrate the knowledge management process into the development process. This method helps software project teams comprehensively and systematically identify and track knowledge management goals as far as possible. With a common framework, an organization is able to exchange knowledge and expertise within itself, which helps to glue the company together; while at the same time ensures that knowledge is shared over time so that the company benefits from past experience. Team members come to a common understanding on how to accumulate knowledge by establishing goals and corresponding solutions to meet the goals, and this consensus and clear vision on knowledge management motivates members to create knowledge and reduce the "gulf" between knowledge creation and application. It was successfully applied in several projects of different companies. The framework helps them establish an initial knowledge and experience repository. Software engineers are able to have more information available than they could understand and apply.**

***Keywords*—Goal Driven; Knowledge Management; Knowledge Management Framework**

I. Introduction

Software engineering is knowledge-intensive work. Therefore, development of software requires knowledge and experience in many areas. Software engineering involves several knowledge types–technical, managerial, domain, corporate, product, and project knowledge [1]. These knowledge and experience help people to decrease software projects' development time and costs, avoid mistakes, reduce rework, and repeat successful processes, as well as increase productivity and the likelihood of further success [1]. They can also help software practitioners make sound decisions under uncertainty and to find better compromises [2]. In one case, effectively exchange of knowledge is the glue that holds a company together [3]. Unfortunately, the reality is that most development teams do not benefit from existing experience as expected and they constantly repeat mistakes even though some individuals in the organization know how to avoid them. Lacking effective Knowledge Management (KM) is one reason of the reality. Therefore, management of knowledge and experience is a challenge for most software organizations. As pointed by Ioana Rus et al., i.e. the challenges and obstacles of KM came from three major sources, the technological issues, the organizational issues and the individual issues [1], [4], [5], respectively.

From the technological perspective, most KM information systems are what so-called closed systems [6]. With closed systems, knowledge and experience are more like documented materials which store answers to questions that might arise during work. Therefore, consumers of the knowledge need to discover the answers when facing specific problems. However, these systems usually contain thousands of pages of unstructured information, as a result, the discovery procedure could be time-consuming and software engineers may lose faith in and patience under tight project schedule. Besides, updating and maintaining existing knowledge and experience will also be a big issue, not to mention obtaining outdated information after long time searching for intended users of the KM systems.

From the organizational perspective, the culture of most industrial organizations usually only encourage finishing current project on time, within budget and with quality. In firms, many resources and much time and effort are required before benefits became visible, as a consequence, current project teams tend to spend little effort and resource to help future projects. For example, organizations with hero culture may encourage individualism rather than cooperative work. Therefore, employee might not be willing to share knowledge and experience at the organizational level. What's worse, few organizations established specific processes for KM and neither maintenance of knowledge and experience nor application of them could be effectively conducted. In this sense, a separated KM mechanism or process might be taken as additional burden on most project teams.

From the individual perspective, development team members might also lack of motivation to accumulate valuable knowledge and experience in current projects. Project managers would rather focus on completing their current project on time than help the next project manager succeed. Therefore, they often consider accumulating knowledge and experience a burden, not mention that "when and how to use the knowledge and experience in future projects is usually uncertain" [1]. For other engineers, they often "do not have time to input or search for knowledge, do not want to give away their knowledge,

978-1-5386-3681-7/17 $31.00 © 2017 IEEE
DOI 10.1109/APSEC.2017.35

and do not want to reuse someone else's knowledge" [1]. Besides, the fast pace of technology evolution often discourages software engineers from analyzing the knowledge they gained during the project, believing that sharing the knowledge in the future will not be useful.

In this paper, we proposed a Goal-Driven Development KM (GDD-KM) framework which featured in its inherent Goal-Driven philosophy and cyclic approach. Goal-Driven philosophy helps to organically structure knowledge and experience since only those knowledge and experience which help to satisfy identified goals in previous projects were captured and organized in the knowledge repository. Meanwhile, the cyclic nature of the GDD-KM helps to enrich, refine and outdate knowledge and experience as necessary. In this sense, knowledge and experience evolve with each cycle. We applied GDD-KM in several projects and helped these teams to accumulate and transfer knowledge and experience effectively and efficiently.

The rest of the paper is structured as follows. Section II provides a brief introduction to the background of this paper, i.e. the related work in KM, GDD project management method and the GDD-KM cycle. Section III describes the application of GDD-KM in a real project. Some empirical evidences were collected and analyzed to verify the effectiveness of GDD-KM. Section IV compares GDD-KM with traditional KM methods and further discusses several limitations on GDD-KM at this stage. The paper is concluded in Section V with the suggestions on future continuous research.

II. Background

A. Related Work in Knowledge Management

Research and practice in knowledge and experience management in software development occurred at different layers and in organizations with different size. Schneider et al. [2] compiled five-year-long experience in building, revising, and improving experience repositories in DaimlerChrysler Research Center. They found that for most knowledge repositories, the contents of the database were unordered and the forms they provided were so sophisticated that they could hardly provide any guidance to its intended users. As Schneider et al. pointed, experiences should not be stored on a dusty shelf, but engineered into best practices and processes that guide user work. Another famous KM implementation came from NASA Software Engineering Laboratory as the first implementation of "Experience Factory" [7]. Liebowitz described this KM implementation based on working experience on NASA and concludes that KM should start small and see what works in a specific environment and that knowledge should be collected during projects, not after their completion [8].

Compared with Experience Factory that requires numerous effort and resources, PMA (PostMortem Analyze) offers a quick and simple way to initiate knowledge management in small- or medium- size software projects. Andreas Birk et al. [9] described an effective solution to address knowledge and experience sharing issue by conducting postmortem with an open atmosphere. They received a lot of positive feedback from PMA participants in different companies. By using systematic postmortem analysis for capturing and reusing experience and improvement suggestions, the teams in one organization increased experience understanding and sharing. However, as Birk et al. [9] said, PMA has been mainly advocated for situations such as completion of large projects, learning from success, or recovering from failure. With popularity of Agile process and practices [10], continuous and quick knowledge and experience evolution and sharing will be more and more important [11], [12].

There also exist numerous successful KM practices. Ramesh studied best practices in 30 organizations and concluded the importance of link knowledge fragment to guarantee traceability (creating and maintaining relationships between objects and people in software development) [13]. This traceability will facilitate successful knowledge transfer and reuse. Komi-Sirviö et al. emphasized the need for addressing local needs, problems, and specific context for KM initiative implementation [14]. Therefore, to make knowledge and experience useful, project-based KM approach will facilitate knowledge collection and delivery better than large scale KM approach at the organizational level.

Through the above-mentioned practice and research, several characteristics of effective KM were identified as the following:

- Standard, flexible structure of knowledge and experience with a clear specification
- To collect knowledge during project development, not after their completion
- To focus on project-based KM approach and to link knowledge fragment to guarantee traceability
- Clear purpose of collecting, managing and using the knowledge

Since most KM approaches might lack these characteristics more or less, we designed a Goal-Driven KM as a build-in mechanism of a project management method, i.e. the Goal-Driven Development (GDD) method.

B. GDD Introduction

Goal-Driven Development (GDD) is a project management method which was designed to improve the traditional project management methods by focusing on balancing and meeting multiple project goals. Practice and evaluation of GDD were described in [15]. A typical cycle of GDD is composed of three primary phases: the launch phase, the development phase and the postmortem phase. Fig. 1 shows the sequence of the GDD method and typical tasks within each phases as well.

1) Practices Supporting KM: Knowledge and experience management was a build-in mechanism of GDD method. Several practices in GDD, especially the practices in launch phase and postmortem phase support KM well.

a) Launch Phase: A launch phase consists of five meetings as depicted in Fig. 1, among which the goals elicitation meeting and the goals implementation solution meeting help to establish the initial version of knowledge and experience repository for project teams.

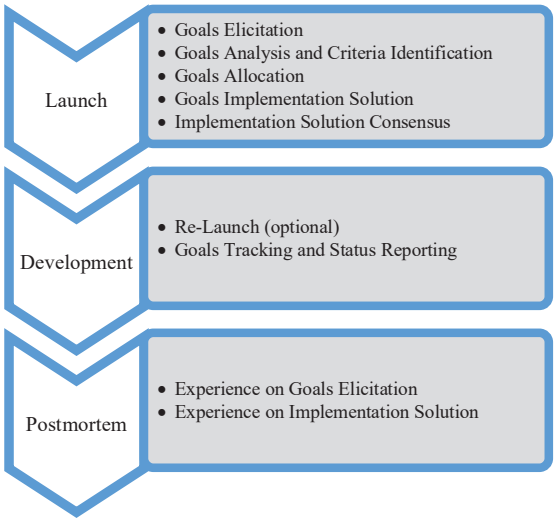

Fig. 1. Process framework of GDD method

Goals Elicitation: In this meeting, the project team members work together to identify two types of goals, i.e. internal and external goals. To elicit goals comprehensively and systematically, a tree-styled diagram (mind-mapping) can be used to facilitate goal identification. The high level goals need to be broken down until the related implementation solutions are applicable and operational to the development team. Typical goals can be grouped into four categories at the top level: product, project, process and team. Fig. 2 depicts an example of goal break-down structure. In practice, many projects are able to group their goals into these four categories.

This tree-styled diagram forms the main skeleton of knowledge and experience repository in GDD-KM which is easy to use in future projects and share among different projects. For example, next projects use this diagram as a goal template to accelerate goal elicitation. Even with different project context, goals are similar in most situations, normally the higher level the more chances to reuse the goal template.

Goals Implementation Solution: To promote full participation, the goals on leaves should be allocated among all the team members according to preference and balanced workload. Therefore, each goal should be assigned a fixed owner, who is responsible to develop the implementation solution and to track the status during development. Note that this does not necessary imply that the owner should devise implementation solution all by her/himself. After that, the whole team should reach consensus on implementation solutions of all the goals on the fine-grained level. Description of the solutions to implement projects goals are the most valuable knowledge and experience for next projects. When all these information has been organized according to the structure of project goals, retrieval of information will be easy. Besides, this goal-solutions structure will also provide software engineers with certain context of application when they create knowledge and experience. Therefore, they have the motivation to accumulate and share individual knowledge and experience.

b) Development Phase: The practices of development phase are relatively simple in GDD, compared with the lifecycle of regular project management process. The goals tracking and status reporting is the only one practice that is compulsory in this phase. Re-Launch is an optional practice according to the tracking results.

The compulsory practice, i.e. goals tracking and status reporting, tracks and reports the status of all the goals identified by the development team. And its main event is team weekly meeting. Before the meeting, the owner of each goals takes responsibility for gathering data and evidence to characterize the status of the goal. Then the team leader will use a tool to summarize the status of all the goals in one chart.

c) Postmortem Phase: In postmortem phase of GDD method, the development team only discusses two questions. 1) How did all the goals identified in the development process support the expectations of all the stakeholders? 2) How did all the implementation solutions help to achieve project goals?

The answer for the first question will help the team find improvement opportunities on goal elicitation; the answer for the second issue will help the team identify effective solutions as the best practices to support project goals. In this way, knowledge and experiences created during the project are well validated for future projects.

C. Cycle of GDD-KM

While GDD usually occurred within project, KM happened across multiple projects. Therefore, from the perspective of KM, GDD-KM should comprise of five typical steps, i.e. knowledge creation, knowledge integration and formation, knowledge dissemination and application, knowledge evolution and knowledge outdated. These five steps forms a cycle which facilities knowledge evolution. Fig. 3 shows the cycle of GDD-KM.

1) Knowledge Creation: In GDD-KM, all the engineers within the project team should participate in the creation of knowledge. However, since knowledge and experience in GDD-KM typically includes set of project goals and related implementation solutions, creation of knowledge may happen through the whole project. For instance, in the launch phase of GDD, all the team members identified project goals and developed solutions to implement these goals. Both the goal set and the solution set are valuable experience and knowledge. Besides, the consensus among the team members created a shared understanding of the knowledge and experience.

2) Knowledge Integration and Formation: The good aspect of full participation of team members in creation of knowledge and experience is that it helps motivate knowledge workers since they are usually also the consumers of the knowledge. Besides, they capture more valuable experience and knowledge since they know what they expect from knowledge and experience repository. However, with ad hoc manner, similar knowledge even redundant knowledge in the knowledge and

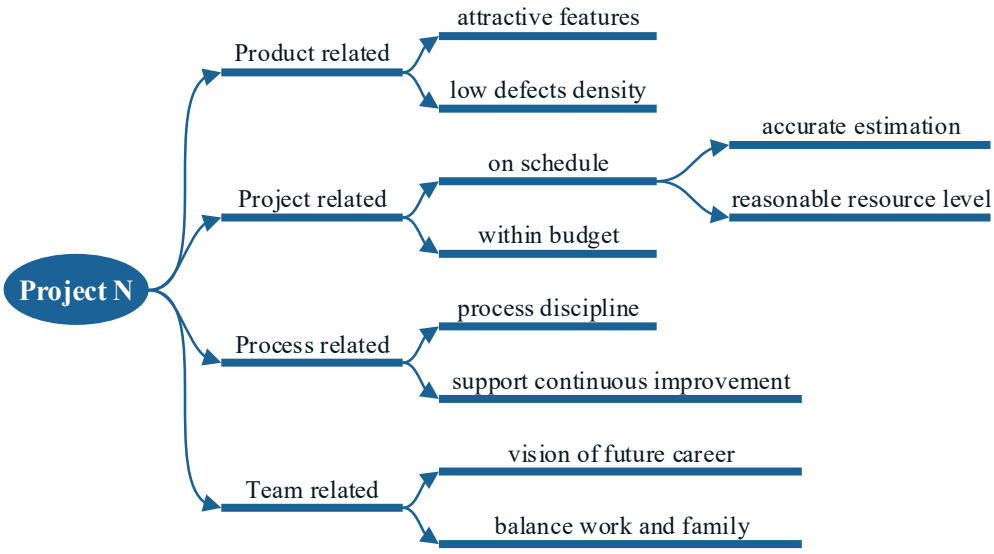

Fig. 2. A breakdown structure of project goals

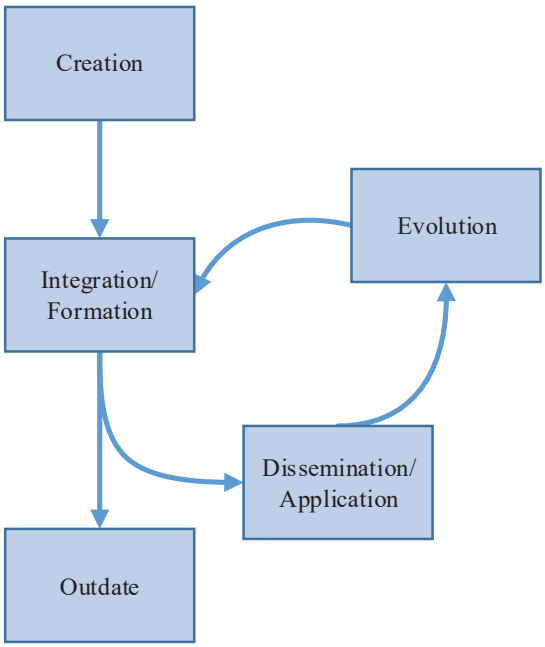

Fig. 3. The cycle of GDD-KM framework

experience repository cannot be avoided. So tasks such as conceptual generalization and representational formalization [6] are also needed in GDD-KM. With conceptual generalization, project team needs extra effort to generalize and abstract context of certain goals and solutions, therefore, the knowledge and experience gained from one project can apply to other projects. With representational formalization, project teams organize knowledge and experience in a way which facilitates access and apply. In GDD-KM, the basic structure of knowledge and experience repository is three layered. Fig. 4 shows the three layers. The top layer is the goal identified by project teams in the organization. In fact, the content of the top level also belongs to the concept of knowledge and experience. With GDD-KM, several goal templates will be defined to record the goals identified in former projects hence to help new projects elicited goals. The intermediate layer is a brief introduction to solutions to address goal on top level. To be pragmatic, the description must be concise and perspicuous. Details of the solutions are described on the bottom layer. On this layer, detailed and concrete guidance are provided to carry out the solutions and to satisfy goals on top level as well.

3) Knowledge Dissemination and Application: GDD-KM is relatively distinct from the traditional KM approaches on its integration of knowledge creation and application. Traditional KM approaches assume that workers perform repetitive and predictable tasks, so they broadcast information (e.g., email), provide searchable databases, disseminate knowledge through classroom training or printed reference documents. These approaches separate learning and working [8]. While in GDD-KM, full participation is highly encouraged in both creation and application of knowledge and experience. Based on similar goal template and corresponding solutions to goals across multiple projects, dissemination of knowledge and experience at organizational level can be feasible. In fact, since goals in

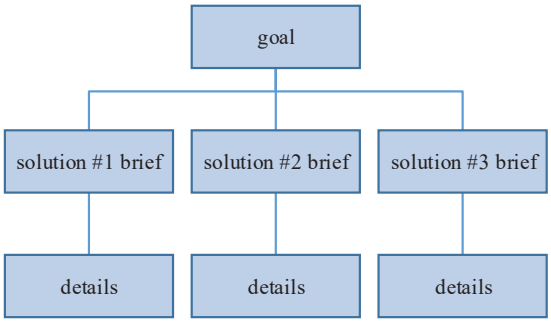

Fig. 4. Basic structure of knowledge and experience repository

GDD were interpreted as internal and external expectations from relevant stakeholders, these expectations were similar except for some technological details. Typical application scenario is in the launch phase of GDD method when project team elicits goals and develops implementation solutions to goals.

a) Goals Elicitation: One challenge a software project team will face at early stage is that how to elicit expectations from relevant stakeholders comprehensively. With GDD-KM, each project team is equipped with a goal template, which provides a basis for further elicitation of project goals. Usually with similar project context, e.g., later development iteration of the same project, nearly no change needs to make to the goal template. Even with different project context, most expectations from stakeholders are addressed by the goal template, hence project team will be at a good position to modify the goal template to meet new expectations.

b) Goals Implementation Solution: With the basic structure of the knowledge and experience repository as depicted in Fig. 4, solutions are attached to certain project goals. Therefore, for those similar goals, there exist solutions in the knowledge and experience repository already. Since most project goals are similar across multiple projects, this helps a lot for goal owners to develop implementation solution.

4) Knowledge Evolution: Knowledge is refined in evolution phase. In this phase, every layer of knowledge in GDD-KM (goal, solution and detail) is refined, updated, and detailed into full conceptual knowledge structures.

Additionally, the focus of KM moves from a specific knowledge transfer, such as how to execute an experiment and reporting the results of an experiment, to a broader knowledge evolution and sharing, such as experiment improvement and how to combine data from different experiments. It means that, after several refining cycles, the knowledge no longer stays in a specific project level, but becomes resource that can be shared and reused in future projects. In fact, this is also an important step to support *Knowledge Integration and Formation* because without properly refining, conceptual generalization might encounter big challenge to handle massive knowledge debris with various contextual information.

5) Knowledge Outdating: Knowledge is context-specific and thus has to change with ever changing environment. However, to be useful the knowledge repository should be kept concise. This requires the knowledge stored to be kept up to date. Appropriate changes need to be undertaken, while outdated knowledge also needs to be discarded.

Different from other KM approaches that blend knowledge outdate process into knowledge update (evolution) procedure [9], in GDD-KM cycle, knowledge outdated acts as a specific step at the same level as knowledge update. With caution, outdated knowledge and experience should be removed from the repository. For example, goal of process discipline can be removed when process discipline was ingrained in the organization culture, solution corresponding to certain goal can be removed while better solutions were identified, and technologies as the solution can be removed when it is substituted or even deprecated, etc.

III. Case Study

We applied GDD-KM in several projects both in industrial environment and academic environment. Results indicated several positive effects of GDD-KM in these projects. In one student project, all the team members conducted effective postmortem and accumulated valuable knowledge and experience. The purpose of this project is to develop a general system with the function of generating schematic map for NYOG (Nanjing Youth Olympic Games). The system should be able to create accurate route, change the relative distance between two adjacent significant features by zooming the image, schematize the route and finally create a schematic map. Using the development method described in [15], the team finished the project and met the requirements. Meanwhile, they also established an initial knowledge and experience repository.

A. Implementation of GDD-KM

Typical steps in a GDD-KM cycle were supported by GDD. Inversely, GDD-KM also helped the team practice GDD method.

1) Knowledge Creation: During launch phase of GDD, the whole project team established the first version of goal set and grouped goals into four categories, in two kinds: external goals (from expectations from senior management or customers) and internal goals (from team members). In this way, a goal template which is similar to Fig. 2 was then formed. This goal template was used in the following several iterations of this project.

After each goal was allocated to a team member, solutions to implement the goal were then developed. As shown in Fig. 5, to meet the quality goal, the team decided to rely on reviews rather than testing to achieve high quality. To direct practice, controlling parameters of the review process were then defined in the details of the solution, for example, review speed, proper duration for a certain review, etc. The whole team then conducted a meeting to reach consensus on both the goal set and corresponding solutions.

Goal	High quality of the final product
Solution	Rely on review rather than testing
Details	Control the review rate (Less than 200 LOC/Hour); Review both detailed design and code by the developer himself/herself; Meanwhile, conduct inspection on core modules;

Fig. 5. Solutions to meet quality goal

2) Knowledge Integration and Formation: During the postmortem phase of GDD, the whole project team reviews the data and information recorded during the development to evaluate the effectiveness of goal elicitation process and implementation solution to address the goals. Some valuable experiences to meet similar goals were then integrated. For instance, in this project, continuous integration and testing was also identified as a solution to address quality goal. Therefore, a new solution was added to the goal of "high quality of the final product".

3) Knowledge Dissemination and Application: In this case, since the project team developed the whole system with multiple iterations, each following iteration had similar context as the first iteration. This provides a good foundation for knowledge dissemination and application. In fact, the project used most of the knowledge and experience gained in this project in other projects. Feedback from team members indicated positive results of GDD-KM (cf. subsection III-B).

4) Knowledge Evolution: In this project, the development team found several useful approaches to get steady review quality, e.g., making a customized review checklist at individual level, using a total different environment than the lab where the team developed the project, etc. Therefore, details of the review-based solution were supplemented and evolved to include new knowledge and experience.

5) Knowledge Outdating: At this early stage of application of GDD-KM, knowledge outdating did not happen in this project.

B. Project Results

This team finished the project on time and met most of the goals identified during launch phase. We conducted an investigation after the project to interview all the project team members. Results showed several positive aspects of GDD and GDD-KM.

1) About Postmortem: Most team members thought that GDD-KM helped them to do better postmortem due to the structured organization of the postmortem process and knowledge and experience framework. Firstly, the goal template provided them the concrete scope to analyze. Secondly, information summarized during postmortem was provided with a certain application context.

2) About Knowledge and Experience Repository: The basic structure of knowledge and repository provided an easy approach to capture and organize valuable information. With GDD-KM, knowledge and experience can also be accumulated and evolved in the repository. Besides, the goals template provides a fast way to retrieval information from the repository.

3) About Continuous Application of Knowledge and Experience: GDD works well in multiple iterations. Therefore, GDD-KM provided a mechanism for fast and continuous application of knowledge and experiences gained in former iterations. This fact not only helped the team to complete the project successfully, but also increased the motivation of the team members to record knowledge and experience. Besides, it also facilitated the verification and validation of knowledge and experience through repeatedly application.

IV. Discussion

Compared to other KM method, GDD-KM features in several aspects, which may support its practical application in software projects. However, there are still several considerations to GDD-KM at this stage, which needs further discussion.

A. Advantages

1) Motivation: The issue of motivation and incentives recurred frequently in discussions of the utility of knowledge management systems and appears to be a critical issue needing further investigation. In many cases, the incentives based on technical approaches only do not appear to be an effective solution, as well as some social and management means. For instance, one company participating in an interview conducted by Hahn, and Subramani [16] created a new role within the organization, the knowledge librarian. The knowledge librarian is responsible for transferring content into the knowledge repository by tagging user submissions with appropriate keywords or meta-data tags. While this solves the problem of increased burden on domain experts and users, this may also create new problems. Since knowledge librarians are not the individuals that actually create the knowledge objects (e.g., documents, reports, videos), even though they may have familiarity with the domain, they may not have an accurate first-hand understanding of the content. Hence, the keywords and meta-data tags appended by the knowledge librarian may be inappropriate and as a result subsequent queries searching for a knowledge resource may not retrieve the right document.

However, GDD-KM methods are based on Victor Vroom's motivation theory [17], which indicates that team members can be highly motivated by clear approaches to achieve success, and that is why each of the team members was required to track a number of goals. The expectancy theory [18] says that individuals have different sets of goals and can be motivated if they believe that there is a positive correlation between efforts and performance, favorable performance will result in a desirable reward, the reward will satisfy an important need, the desire to satisfy the need is strong enough to make the effort worthwhile. Since team members all agree to track these goals and they are all responsible for several goals, they are more likely to participate in the knowledge accumulation.

2) Full-Team Participation: Learning is a fundamental part of KM because employees must internalize (learn) shared knowledge before they can use it to perform specific tasks. Individuals primarily learn from each other, by doing, and through self study. Knowledge also spreads from individuals to groups, and throughout organizations and industries. KM aims to elevate individual knowledge to the organizational level by capturing and sharing individual knowledge and turning it into knowledge the organization can access. Individuals eventually perform tasks to achieve organizational-level goals. Therefore, the iterative knowledge processing and learning activities at the individual level are of utmost importance. As Peter M. Senge says, "Organizations learn only through individuals who learn. Individual learning does not guarantee organizational learning. But without it no organizational learning occurs." [19].

Full participation and team decision are highly recommended in GDD-KM. By means of be responsible for several goals, team members all take part in the management and are more likely to have a shared understanding of the definition, structure of their knowledge management system, as a result, makes it much more easier to find some knowledge they expects.

3) Integration of Development Management and Knowledge Management: Organizations constantly need to decrease software projects' development time and costs. Avoiding mistakes reduces rework; repeating successful processes increases productivity and the likelihood of further success. So, organizations need to apply process knowledge gained in previous projects to future projects. Unfortunately, the reality is that development teams do not benefit from existing experience and they repeat mistakes even though some individuals in the organization know how to avoid them. Project team members acquire valuable individual experience with each projectthe organization and individuals could gain much more if they could share this knowledge.

Besides, in software development, every person involved constantly makes technical or managerial decisions. Most of the time, team members make decisions based on personal knowledge and experience or knowledge gained using informal contacts. This is feasible in small organizations, but as organizations grow and handle a larger volume of information, this process becomes inefficient. Large organizations cannot rely on informal sharing of employees' personal knowledge. Individual knowledge must be shared and leveraged at project and organization levels. Organizations need to define processes for sharing knowledge so that employees throughout the organization can make correct decisions.

Firstly, every organization has its own policies, practices, and culture, which are not only technical but also managerial and administrative. New developers in an organization need knowledge about the existing software base and local programming conventions. Unfortunately, such knowledge typically exists as organizational folklore. Experienced developers often disseminate it to inexperienced developers through ad hoc informal meetings; consequently, not everyone has access to the knowledge they need. Passing knowledge informally is an important aspect of a knowledge-sharing culture that should be encouraged. Formal knowledge capturing and sharing ensures that all employees can access it.

Secondly, software organizations depend heavily on knowledgeable employees because they are critical to the project's success. However, accessing these people can be difficult. Software developers apply just as much effort and attention determining whom to contact in an organization as they do getting the job done. These knowledgeable people are also very mobile. When a person with critical knowledge suddenly leaves an organization, it creates severe knowledge gapsbut probably no one in the organization is even aware of what knowledge they lost. Knowing what employees know is necessary for organizations to create a strategy for preventing valuable knowledge from disappearing. Knowing who has what knowledge is also a requirement for efficiently staffing projects, identifying training needs, and matching employees with training offers.

Finally, software development is a group activity. Group members are often geographically scattered and work in different time zones. Nonetheless, they must communicate, collaborate, and coordinate. Communication in software engineering is often related to knowledge transfer. Collaboration is related to mutual sharing of knowledge. Group members can coordinate independently of time and space if they can easily access their work artifacts. So, group members need a way to collaborate and share knowledge independently of time and space.

As Michael H. Zack mentioned [20], the degree to which knowledge is an integral part of a company is defined not by what the company sells but by what it does and how it is organized. Many companies in recent years have redefined their mission from one based on selling traditional products to one based on exploiting knowledge.

With a consensus and a framework of knowledge management, the knowledge can be easily transferred and understood by team members as well as the other relevant crews in the company. Many companies nowadays like one of the world's largest suppliers of cement realized that the exchange of knowledge is the glue that holds the company together.

Products and services are only what are visible or tangible to customers–they are the tip of the iceberg, most of what enables a company to produce anything lays below the surface, hidden within the so-called invisible assets of the organization–its knowledge about what it does, how it does it and why.

GDD-KM ensures that knowledge is shared over time so that the company benefits from past experience to make it possible for people from various parts of organization to find one another and collaborate to create new knowledge and to provide opportunities and incentive for experiment and learning. Moreover, the goals in our framework are stated in short names of several words, one sentence's comments and a detail description. We use this structure to make the query process faster and easier.

B. Limitation of GDD-KM

GDD-KM provides a pragmatic framework for knowledge and experience management. However, there still exist several limitations at this stage.

Firstly, GDD-KM is a build-in mechanism of GDD method. Therefore, it highly relies on GDD. Project teams must use GDD to manage their projects so as to provide the data needed for GDD-KM, e.g., the goals template, the solutions attached to certain goals, etc.

Secondly, GDD-KM is more suitable to handle what so-called codification [21] strategy to manage knowledge and experience. Within this approach, the role of KM is to support the storage and retrieval of explicit documented knowledge by people throughout the organization as and when required. However, there exists another strategy called personalization [21]. Within this strategy, KM is used to extend interpersonal networks and the ability to connect and communicate with one another [21]. We believe much can be gained by more researches on how to mix the Codification and Personalization via GDD-KM methods.

Thirdly, for pragmatic purpose, knowledge representation in GDD-KM is based on natural language at this stage. Redundant information and imprecise description cannot be totally avoided. With increased size of the knowledge and experience repository, it will be more and more difficult to conduct knowledge integration and formation.

C. Research Consideration

The research in this paper indicated several positive results of GDD-KM. However, there still exist several considerations while applying GDD-KM in practice.

Firstly, the project described in this paper is a student's contest project. The difference on main purpose between industrial projects and this academic project may affect the results. For instance, for industrial projects, to finish project on time, within budget and with high quality may constitute the top goals; meanwhile, for contest projects, usually to finish the project on time, with attractive GUI may become the top goals. However, the project in this research was developed in summer school when students worked just as full-time professionals. Research conducted by Runeson et al. [22], [23] indicated that with similar environment, similar improvements trends can be identified both in industrial projects and academic projects. In this sense, more reports on GDD-KM with industrial contexts are necessary.

Secondly, considering the duration and staff size, the projects we applied GDD-KM are not big projects. When the scale of the project increases, the number of goals and related solutions will also increase as well. Therefore, it will be much more difficult manage knowledge and experience. The *Knowledge Integration and Formation* step in GDD-KM cycle is designed to address this issue, however, it requires enough knowledge on software development and rich experience on GDD.

V. Conclusion

Knowledge workers as software engineers now have more information available than they could understand and apply, while on the other hand, finding information relevant to the task at hand is becoming increasingly critical. To address information overload, KM approaches must provide the information workers need, when they need it. The tree-structure goals and three layer mode of representation in GDD-KM provide an organized guidance for users to create, integrate and apply the knowledge to future projects. Besides, thanks to the shared goal template in different projects within organizations, the corresponding solutions to these goals can also be shared by these teams, which improve the knowledge transfer in different projects. Moreover, the software development process (i.e. GDD) and the knowledge management process (i.e., GDD-KM) are seamlessly integrated, which may facilitate its adoption in practices.

Another contribution of GDD-KM is to help establish an initial knowledge and experience repository according to a common goal template. Schneider et al. mentioned that "Be Specific" and "User Guidance" should be considered as two significant success factors [2]. It means that, for one thing, "Experience Repository" should be well-organized; for another, "Repository" should be organized as simple and flexible as possible to facilitate the usage of knowledge to the full extent.

Our work is a worthy attempt to manage knowledge and experience in software engineering pragmatically. There still exist several interesting issues which need future research, for example:

1) *How to combine Codification with Personalization in GDD-KM?* As Anthony et al. [24] claimed that the field is moving from first to second generation, KM is characterized by knowing-in-action. This means Personalization became more and more important to make KM useful. Therefore, more research needs to be conducted to combine both strategies in GDD-KM.
2) *How to handle the scale issue?* With increased project scale, KM will be more and more difficult and ineffective due to mass information. Therefore, to design new approaches to represent knowledge and experience and new methods to store and retrieve useful knowledge in GDD-KM is a promising research direction.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant No.61572251).

References

[1] I. Rus and M. Lindvall, "Knowledge management in software engineering," *IEEE Software*, vol. 19, no. 3, p. 26, 1 May 2002.
[2] K. Schneider and J.-P. von Hunnius, "Effective experience repositories for software engineering," in *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*. Washington, DC, USA: IEEE Computer Society, 3 May 2003, pp. 534–539.
[3] M. H. Zack, "Rethinking the knowledge-based organization," *MIT sloan management review*, vol. 44, no. 4, p. 67, 1 July 2003.

[4] M. Alavi and D. E. Leidner, "Knowledge management systems: Issues, challenges, and benefits," *Commun. AIS*, vol. 1, no. 2es, February 1999.
[5] 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, 31 December 2016.
[6] G. Fischer and J. Otswald, "Knowledge management: Problems, promises, realities, and challenges," *IEEE Intelligent Systems*, vol. 16, no. 1, pp. 60–72, January 2001.
[7] F. Houdek, K. Schneider, and E. Wieser, "Establishing experience factories at Daimler-Benz: An experience report," in *Proceedings of the 20th International Conference on Software Engineering (ICSE '98)*. IEEE, 19 April 1998, pp. 443–447.
[8] J. Liebowitz, "A look at NASA goddard space flight center's knowledge management initiatives," *IEEE Software*, vol. 19, no. 3, pp. 40–42, May 2002.
[9] A. Birk, T. Dingsøyr, and T. Stålhane, "Postmortem: Never leave a project without it," *IEEE Software*, vol. 19, no. 3, pp. 43–45, May 2002.
[10] M. Desmond, "Developers mix and match agile approaches," 1 March 2010. [Online]. Available: http://adtmag.com/articles/2010/03/01/developers-mix-and-match-agile-approaches.aspx
[11] V. Santos, A. Goldman, and C. R. B. de Souza, "Fostering effective inter-team knowledge sharing in agile software development," *Empirical Software Engineering*, vol. 20, no. 4, pp. 1006–1051, 1 August 2015.
[12] M. A. Razzak and D. mite, "Knowledge management in globally distributed agile projects–lesson learned," in *2015 IEEE 10th International Conference on Global Software Engineering (ICGSE '15)*. IEEE, 13 July 2015, pp. 81–89.
[13] B. Ramesh, "Process knowledge management with traceability," *IEEE Software*, vol. 19, no. 3, pp. 50–52, May 2002.
[14] S. Komi-Sirviö, A. Mäntyniemi, and V. Seppänen, "Toward a practical solution for capturing knowledge for software projects," *IEEE Software*, vol. 19, no. 3, pp. 60–62, May 2002.
[15] G. Rong, D. Shao, H. Zhang, and J. Li, "Goal-driven development method for managing embedded system projects: An industrial experience report," in *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM '11)*. Washington, DC, USA: IEEE Computer Society, 22 September 2011, pp. 414–423.
[16] J. Hahn and M. R. Subramani, "A framework of knowledge management systems: Issues and challenges for theory and practice," in *Proceedings of the Twenty First International Conference on Information Systems (ICIS '00)*. Atlanta, GA, USA: Association for Information Systems, 10 December 2000, pp. 302–312.
[17] V. H. Vroom and E. L. Deci, *Management and Motivation*. Penguin, 1989.
[18] W. Van Eerde and H. Thierry, "Vroom's expectancy models and work-related criteria: A meta-analysis," *Journal of applied psychology*, vol. 81, no. 5, p. 575, October 1996.
[19] P. M. Senge, *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization*. Crown Business, 2014.
[20] M. H. Zack, "Developing a knowledge strategy," *California Management Review*, vol. 41, no. 3, pp. 125–145, April 1999.
[21] D. Torgeir and R. Conradi, "A survey of case studies of the use of knowledge management in software engineering," *International journal of software engineering and knowledge engineering*, vol. 12, no. 4, pp. 391–414, August 2002.
[22] P. Runeson, "Using students as experiment subjects–an analysis on graduate and freshmen student data," in *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering*, 8 April 2003, pp. 95–102.
[23] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects—a comparative study of students and professionals in lead-time impact assessment," *Empirical Software Engineering*, vol. 5, no. 3, pp. 201–214, 1 November 2000.
[24] A. F. Buono and F. Poulfelt, *Challenges and Issues in Knowledge Management*. IAP, 2005.