Characterizing the Utilization of Knowledge Transferred in Software Architecture Development

Salfarina, A, Marzanah, A. J.Sazly, A.
Faculty of Computer Science and Information Technology, University Putra Malaysia
43400 Serdang Selangor, MALAYSIA.
sal@fsktm.upm.edu.my, marzanah@putra.upm.edu.my

ABSTRACT

Knowledge transfer (KT) is crucially essential for making design decisions in software architecture development, where many factors and inputs need to be carefully considered. However, not much is known about KT in software architecture development, a setting that is knowledge intensive. Unclear flow of KT is identified as one problem seeking for immediate delineation. This include to what extent the knowledge transferred is utilized among those involved. The main purpose of this paper is therefore to provide an understanding of how knowledge transferred can be characterized in terms of its usage during the development of software architecture, which normally involves the analysts and software architects. Fifteen items of questionnaire derived from all the main activities in developing software architecture were asked to 30 industrial experts. The primary contribution of this study is a technique to characterize the utilization of knowledge transferred in order to determine the extent of knowledge use, which can prove the establishment of an effective KT. It further highlights the emphasis of putting the knowledge into action as an important indicator or prerequisite of KT.

Keywords: Knowledge transfer (KT), knowledge utilization, use, characterization, non-collocated software architecture.

1. INTRODUCTION

Operationally, the measurement of utilization has centered almost exclusively on the instrumental use of knowledge that is the contributions of knowledge to the decisions of practitioners and decision makers [1]. In software architecture development, software architects make design decisions based on the requirements analysis done by the analysts. Obviously, they require all necessary inputs and knowledge before they can reason out these decisions. [2] Highlights information and knowledge processing before one can reach to knowledge utilization. [3] Further add that instrumental use or behavioral use of knowledge is where knowledge is adapted when used to fit the users’ specific needs. This is indeed how a typical scenario is portrayed while developing software architecture. The question is what do they do with their existing knowledge and how others’ knowledge can be adapted to accomplish the right decisions? Our intention is to characterize the knowledge use among those involved in software architecture development. We believe our attempt is significant as it draws attention to the essential of KT, which is the application or utilization of knowledge being transferred. The outcome or result of knowledge utilization is greatly displayed by the decisions made including enhancement of their knowledge.

In the following sections, the current body of literature review in this study is explained; the characterization of knowledge utilization is introduced and discussed, and finally a conclusion.

2. LITERATURE REVIEW

2.1 Knowledge Transfer

KT is the dissemination of knowledge from one individual or group to another within the organization. It may be purposely transferred, or it may occur as an unintended outcome of other activities [4]. As asserted by [5] “…knowledge are best learned through exposure to and experience…” This is further supported by [6] where according to her, KT implies that each individual or group or organizational unit need not learn from scratch but can rather learn from the experiences of others. Therefore in this paper, we adopted the definition of KT as the process through which one unit learns from the experiences of ([7] and [8]). From our perspective, KT is about the integration of knowledge and experience between people from various backgrounds and expertise. This is in line with the knowledge intensive perseverance in software architecture development, which demands such integration. These people need not only sharing but also learning from each others’ experience to ensure that they can accomplish their tasks. It is also believed that the definition of KT must cover the use of knowledge on the part of the receiver ([9] and [8]) and not simply by sharing of the knowledge between units. This is particularly important to distinct the overlapping terms between KT and just knowledge sharing, and also makes it easier to verify that KT has occurred by investigating those cases involving use, which can be observed and measured [8]. Given all these definitions, we can foresee that the role KT plays is critical to ensure the continuity of success to the organization, and also to the capability development of those involved in KT.
Utilization of exchanged knowledge corresponds to the evaluation that is the anticipated effects as a result of transferring the knowledge. It signifies the importance of knowledge utilization as a prime evident of KT occurrence. Knowledge is taken to be transferred when learning takes place and when the recipient understands the intricacies and implications associated with that knowledge or she can apply it ([10] and [8]). [9] highlight this in their definition of KT, where unless knowledge is absorbed, it is not transferred; merely making knowledge available does not equate to transfer. [11] support this argument by claiming that KT is evident when experience acquired in one unit affects another. [12] shares similar opinion by stressing that even transmission and absorption together have no useful value if the new knowledge does not lead to some change in behavior. In the study by [13], the extent to which the task-performing unit needs to learn from others is one of the key characteristics that are likely to influence whether utilizing the firm’s knowledge resources enhances or undermines task performance. They have also given a simple definition to knowledge utilization, which is the process of obtaining and using knowledge from documents and other people. With knowledge-integration, team members work collaboratively in a way that encourages ongoing, constructive dialogue so that the valuable resources within the team can be effectively utilized for team performance [14].

2.2 Software Architecture

The definition of software architecture includes all the usual technical activities associated with design: understanding requirements and qualities; extracting architecturally significant requirements; making choices; synthesizing a solution; exploring alternatives and validating them [15]. In software development process, software architecture is generally a part of preliminary design in the design phase. It includes negotiating and balancing of functional and quality requirements on one hand, and possible solutions on the other hand. This means requirements development and software architecture are not subsequent phases that are more or less strictly separated, but instead they are heavily intertwined. There are many reasons describing the importance of software architecture phase in software development process. Firstly, it is a vehicle for communication among stakeholders. Software architecture is a global, often graphic, description that can be communicated to the customers, end users, designers and so on. By developing scenarios of anticipated use, relevant quality aspects can be analyzed and discussed with various stakeholders. The software architecture also supports communication during development. This is consistent with the empirical evidence by [15], where the architecture almost always exists as knowledge of people applied and communicated answering situated questions and problems. Secondly, it captures early design decisions. In software architecture, the global structure of the system has been decided upon, through the explicit assignment of functionality to components of the architecture. These early design decisions are important since their ramifications are felt in all subsequent phases. It is therefore paramount to assess the quality at the earliest possible moment. Thirdly, architecture is the primary carrier of a software system's quality attributes such as performance or reliability. The right architecture is the linchpin for software project accomplishment whereby the wrong one is a recipe for guaranteed disaster.

2.3 The Importance of KT in Software Architecture Development

It is agreed that both analysts and software architects play important roles in the successful software architecture, and that the transfer of knowledge is important in the software architecture development. However, not much is known about KT between analysts and software architects — a setting that is very much knowledge intensive. Initially, the analyst primarily possesses business knowledge, whereas the software architect primarily possesses technical (including architectural) knowledge [16]. KT between these two teams invites an intriguing intention for discovery of the flow and nature of the transfer considering the lack of its descriptions in the literature. The integration of initial knowledge possessed by these teams is seen as a must. More importantly, there are other elements surrounding this process (of KT) alongside the constraints of the environment that need to be taken into consideration.

Software architecture development is where knowledge integration mostly occurs compared to other phases in software development life cycles. It is the encounter of two most highlighted roles for developing software architecture – the analyst and software architect teams. Both teams are specialized in different types of knowledge, background and capabilities. Although they are assigned with different job responsibility, they are highly dependent on each other. Software architect needs input from the analyst and vice versa to complete each other’s objective. But certainly the dependency that exists between them is not only limited to the need for delivering their tasks to develop software architecture. Instead, at the same time it initiates the urgency to learn about each other’s expertise, knowledge and experience, thus creating the opportunity for KT. As a result, they create new knowledge and increase their own knowledge possession. Through this communication, the software engineer who shares his knowledge also updates his knowledge [15]. Now that they are well aware on how and where to locate and access expertise, they are well understood about each other’s accountability, the process of developing the software architecture will eventually become much smoother, faster and less problematic.

It seems rightly emphasized to rationalize the importance of KT since software architecture development
acts as a vehicle for communication among those who are involved. As a blue print that describes the whole software/system, it is a necessity for it to be effectively delivered and communicated. KT determines this by ensuring that the software architecture produced is the outcome of integration of knowledge particularly between the analysts and software architects. Without KT, the development of software architecture might be imprecise and does not provide adequate information to proceed to the next phase of development.

Making decision is never an easy task. Software architect is held accountable for making early design decisions during software architecture development. These decisions are partly made based on the input and requirements provided by the analysts. KT is crucially essential as for making these design decisions, many factors and inputs need to be carefully considered and accounted. Both teams must provide as much information as possible to ensure that they can come out with the best decision for software design and at the same time ensuring that the user requirements are fulfilled.

3. CHARACTERIZING THE UTILIZATION OF KNOWLEDGE TRANSFERRED IN SOFTWARE ARCHITECTURE DEVELOPMENT

There are few steps in this attempt. The first step is to identify the areas of knowledge used and exchanged in software architecture development. The second step highlights the interdependency between those involved while developing software architecture. Next is to construct a questionnaire based on the main activities in software architecture development and finally conduct the interviews with selected industrial experts.

a. Identify the areas of knowledge used and exchanged

Based from the literature review, we managed to compile four distinct areas of knowledge that are most likely used in software development, into a list that comprised of technical, application domain, project management and people knowledge. Technical knowledge area encompasses a breadth of knowledge: programming, problem solving strategies, code testing & debugging, development knowledge and skills, architecture concepts & techniques, detailed design, design constraints, specific and general technologies & platforms, software development methods and specification techniques & languages, software design principles, abstractions of design/code as schemas or plans, and design techniques & tools, ([17], [18], [19], [20], [15], [21], [22], [23], [24], [25], [26], [27]). Application domain knowledge area concerns about the specific system to which the software pertains, customers’ business process, client operations, business rules, stakeholders’ needs, as well as the customers’ business objectives ([17], [19], [15], [23], [21], [24]). While project management knowledge deals with planning, staffing, managing and leading a project ([20], [15], [26]), people knowledge on the other hand, accounts the knowledge about leadership, teamwork, communication, negotiation, accepting direction, mentoring and consulting [28].

Software analyst and software architect each possess different set of knowledge and expertise although working in similar domain. While the area of knowledge for software analyst is primarily focused on requirement gathering and analysis, software architect in another hand majors in knowledge of software architecture. To produce software architecture, there is a necessity for an integration of this knowledge. This premise is consistent with the views of [29], who state that the "difference in knowledge expertise between team of experts … drive knowledge transfer". [14] Further support the premise when according to them, outcomes are better when team members are aware of the knowledge others hold. Based on the literature, technical and domain knowledge are most likely to be exchanged and applied when developing the software architecture between these two roles. Other knowledge that might be involved in the transfer activity includes project management and people knowledge.

b. Highlight the interdependency between those involved

In our study, the interdependency refers to the linkage between the analyst and software architect roles that play both sender and receiver in KT. The interdependency explains the need to communicate with each other to ensure quality deliverables can be produced. For example, software architect can be guaranteed to receive complete document of software requirement specifications so that quality software architecture can be produced.

c. Construct a questionnaire

According to [28], software architecture development process consists of five primary activities including creating business case, understanding the requirements, creating and selecting architecture, documenting and communicating the architecture, and analyzing and evaluating the architecture. We construct 15 items concerning the application of related knowledge into each possible step-by-step activity in software architecture development. Every item asked was constructed in a way it tells where the participant gain the knowledge from, and how does the knowledge being put into use to accommodate the activities involved.
d. Conduct the interviews with the experts

30 industrial experts ranging from analysts, software architects to project managers from different software development organizations were selected and interviewed. All interviews were conducted in semi-structured form according to the participants’ own time and venue preferences. Each session took about 30 minutes. The data collected were then analyzed and presented as in the following.

THE CHARACTERIZATION

As anticipated, majority of the participants have successfully characterized the extent of their knowledge utilization. As shown in Table 1.0, 100% of the participants agree and strongly agree that they perform all of the listed items regarding knowledge utilization. This suggests that they have engaged in KT and prove that they have actually applied the knowledge they gained into their tasks. This is consistent with the requirement or prerequisite of effective KT that emphasizes putting the knowledge into action and not merely knowledge transferring and receiving situation.

Table 1: Characterization of knowledge utilization

<table>
<thead>
<tr>
<th>Items</th>
<th>Frequency (and percentage %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Somehow agree</td>
</tr>
<tr>
<td>Using the knowledge gained from the mentoring session held prior to starting the project, we analyze software requirements.</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>We held regular meetings and discussions for both teams in order to ensure we understand business and customer needs before development begins.</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>We capture software specifications from business requirements described by the clients through brainstorming session.</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Using our architectural and design knowledge, we articulate and refine architectural requirements.</td>
<td>11 (36.7%)</td>
</tr>
<tr>
<td>Using our knowledge in software development methods, we document the defined requirements to produce Software Requirement Specification (SRS).</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Through several meetings and progress reviews, we get input on needs to evolve and improve the architecture.</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>We create/draw the initial architecture based on an analysis of the given requirements.</td>
<td>3 (10%)</td>
</tr>
<tr>
<td>We often use reference architecture and make some adjustments to save time on architectural decisions.</td>
<td>10 (33.3%)</td>
</tr>
<tr>
<td>We make design decisions based on mutual agreement with the other team.</td>
<td>4 (13.3%)</td>
</tr>
<tr>
<td>Using our architectural and design knowledge, we identify the style and articulate the principles and key mechanisms of the architecture partitioning the system.</td>
<td>9 (30%)</td>
</tr>
<tr>
<td>We define how the various components fit together.</td>
<td>3 (10%)</td>
</tr>
<tr>
<td>We evaluate the architecture through various means including prototyping, reviews, and assessments.</td>
<td>5 (16.7%)</td>
</tr>
<tr>
<td>We do trade-off analysis on the design through active discussions with the business/software analyst team.</td>
<td>4 (13.3%)</td>
</tr>
<tr>
<td>Using the application domain knowledge gained from the early phase of requirement analysis, we document the domains for which the system/software will be built.</td>
<td>2 (6.7%)</td>
</tr>
<tr>
<td>We prepare architectural documents and deliver presentations to the stakeholders and other development teams.</td>
<td>0 (0%)</td>
</tr>
</tbody>
</table>
We also found that although both teams produce different deliverables, their tasks are overlapping dependent by nature. This simply means that there are tasks involving both teams that rely on their capability to make mutual decision, “… in order to reach a consensus regarding the multiple interpretations of the software requirements … and clarify any existing instances of role ambiguity” [30]. They are not just sequentially dependent but they corroborate each other to accomplish their tasks. For example, as commented by one of the participants:

“As a software architect, although I am not directly involved in requirements gathering, I work together with the SA (analyst) to articulate and refine architectural requirements. This is important to ensure that the architecture fulfills the requirements and clients’ expectations.”

4. CONCLUSIONS

We propose a technique that enables characterizing the utilization of knowledge transferred. The purpose is to provide an evident of the KT occurrence, which in this case refers to KT between analyst and software architect teams during software architecture development.

Recall that we choose to define KT as learning from the experience of others. It is worth noting that every activity in the software architecture development involves collaboration of both analyst and software architect teams. The task specified for each activity either requires the application of knowledge obtained from previous engagement with other people/team or necessarily demand for participation from other people/team for their input, view and agreement on certain issues. This has therefore strengthened the fact that KT in software architecture development does not only address the utilization of knowledge but put the emphasis in the essentials of learning from others and their experiences.

REFERENCES


