A Model for Characterizing Knowledge Utilization in Verifying Successful Knowledge Transfer

Salfarina Abdullah[1], Hazlina Hamdan[1], Marzanah Abdul Jabar[1] and Sazly Anuar[2]

[1]*Universiti Putra Malaysia, Malaysia, {salfarina@upm.edu.my, hazlina@upm.edu.my, marzanah@upm.edu.my}*
[2]*Universiti Kuala Lumpur Malaysia France Institute, Malaysia, {sazly@unikl.edu.my}*

ABSTRACT

Software architecture development is a creative process that requires integration of different knowledge expertise and functionalities hence, open up wide opportunities for those involved to learn from each other. Nevertheless, many assertions have been made about the flow of knowledge transfer being unclear and abstruse therefore led to failure in maximizing knowledge transfer benefits. We respond to this assertions by proposing a model for characterizing the utilization of knowledge in order to verify the occurrence of successful knowledge transfer. The model is derived based on PKAMI, a strategy we invented that comprises of 5 key steps. We further presented a simple case study to show our implementation of the model proposed. The significance of this research is that the model will be able to assist all software practitioners to verify their engagement in knowledge transfer as well as to better strategize on improving and keep on producing quality software project deliverables. We are also aiming at elevating the essence of knowledge utilization by encouraging those involved in development to find ways and opportunities to learn from others' experiences.

Keywords: knowledge utilization, knowledge transfer, software architecture development.

I INTRODUCTION

.Software architecture development is where knowledge integration mostly occurs compared to other phases in software development life cycle. 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 confined to the need for delivering their tasks but at the same time, it initiates the insistence to learn about each other's expertise, knowledge and experience, thus creating the opportunity for knowledge transfer (KT) (Abdullah, S. et al. 2012).

However, having to work under tight budget and time constraint, these KT opportunities often left unattended. When knowledge does not flow among project teams within an organization, resources are wasted particularly loss of important knowledge (Porrawatpreyakorn et al. 2009; Polyaninanova, T. 2011). The tendency of reinventing the wheel, and proposing poor solutions and decisions will eventually lead to software project failures. Yet, many claimed to have engaged in KT. KT is deemed successful only if knowledge transferred is utilized. Hence, this raises up a question: How can we actually verify that KT has occured? Therefore the aims of this paper are: 1) to propose a model for characterizing the utilization of knowledge in software architecture development and 2) to elevate the essence of knowledge utilization as a determinant to successful knowledge transfer.

II THE LINKAGE BETWEEN SOFTWARE ARCHITECTURE DEVELOPMENT, KNOWLEDGE TRANSFER AND UTILIZATION OF KNOWLEDGE

A. Software Architecture Development

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 (Unphon and Dittrich, 2010). 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 development 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. 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.

B. The Importance of Knowledge Transfer in Software Architecture Development

From our research 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 environment 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. This is consistent with the empirical evidence by Unphon and Dittrich (2010), where the architecture almost always exists as knowledge of people applied and communicated answering situated questions and problems.

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.

C. Knowledge Utilization As A Determinant of Knowledge Transfer

Henninger (2001) said that the main problem in knowledge management (KM) is neither capture nor store knowledge, but use them to support the execution of current activities. The use of knowledge or knowledge utilization refers to the action of making the knowledge useful for the knowledge receiver to accomplish his goal and produce the right decision when needed. Knowledge must be used as the basis for the development of new knowledge through integration, innovation, creation, and extension of the existing knowledge basis and should still be used as a basis for decision making (Gonzalez and Martins, 2017). This is particularly true as making the right decision determines successful SD project. To understand software development and its practices, it is very important to not only understand the software but also appreciate software developers. According to Paivarinta and Smolander (2014), software developers must continuously reflect their knowledge on software development and use and build local theories of their own and their teams' actions in developing software. This has shown that knowledge is indeed inseparable from action. The practices in software development demand such knowledge utilization to ensure software developers able to produce the intended and quality project deliverables.

There have been many cases however that demonstrated poor quality of project deliverables in SD. These include incomplete software requirement specification, and incorrect design decision. We believe incapacity to effectively utilized the knowledge has given major contribution to such cases. On the other hand, when knowledge is transferred effectively, the effect will not only benefit the receiver, but also the environment where KT occurs. The utilization of knowledge however does not only influenced by a single element but it also relies on the knowledge itself, the medium used for transferring the knowledge, as well as credibility of the receiver. For our research purpose, the utilization of knowldge or knowledge utilization is defined as the action of putting the knowledge received into use and translating it into useful and effective process, as well as project deliverables.

The utilization of exchanged knowledge corresponds to the anticipated effects as one of the key elements of KT. 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 (Argote 1999; Darr and Kurtzberg 2000). Davenport and prusak 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. Argote et al. (2003) support this argument by claiming that kt is evident when experience acquired in one unit affects another. Abou-zeid (2008) 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 Haas and Hansen (2005), 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. 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 (Gardner et al. 2011).

III THE PROPOSED MODEL FOR CHARACTERIZING THE KNOWLEDGE UTILIZATION

Responding to the research question posit in the beginning, we propose a model for characterizing the knowledge utilization in order to verify successful KT. It will also be used to determine the extent of knowledge use after transferring completed. Our design strategy consists of the following 5 steps, which is named 'PKAMI':

- Identify the particular software Process where KT is expected to occur
- Determine the general and specific Knowledge areas used involved during the development of the software architecture
- Determine primary and secondary Activites involved in the development of the software architecture
- Determine the Medium used in facilitating the software architecture development activites
- Construct the questionnaire Items according to the knowledge utilization scales and tivities in the software architecture development.

A scale consists of different stages of knowledge utilization developed by Knott and Wildavsky (1980) has been referred and applied by many researchers with interest in knowledge use (see Table1.0). The scale has six stages of utilization: 1) Reception, 2) Cognition, 3) Discussion, 4) Reference, 5) Adoption, and finally 6) Influence. Using this scale, knowledge in software development can be characterized by identifying the extent of knowledge use. Based on the table below, the first two stages neither not exactly reflecting the use of knowledge nor explicitly affecting the receiver and its environment, therefore they are categorized as No KT. Stage 3 onwards on the other hand, implies the position of knowledge into meaningful use, thus Yes KT.

Table 1. Knowledge Utilization Stages (KUS)

Stage	Description
Reception	Information is received; within reach
Cognition	The information is read and understood
Reference	The information changes the way the person views the topic area or situation
Effort	The information influences action
Adoption	The information influences outcome
Implementation	The information becomes incorporated into practice
Impact	The information yields tangible benefits

Our approach in deriving this model relies on our knowledge about the first four steps of PKAMI. Based on that knowledge, we construct the questionnaire items concerning the possible application of related knowledge into each possible step-by-step activities in software architecture development. It is important to note that the questionnaire items are developed are strictly based on the process currently being studied in order to verify successful KT. Every item is constructed in a way it can tell where the participant gain the knowledge from, and how does the knowledge being put into use to accommodate the activities involved. This is the part where we start characterizing the knowledge use.

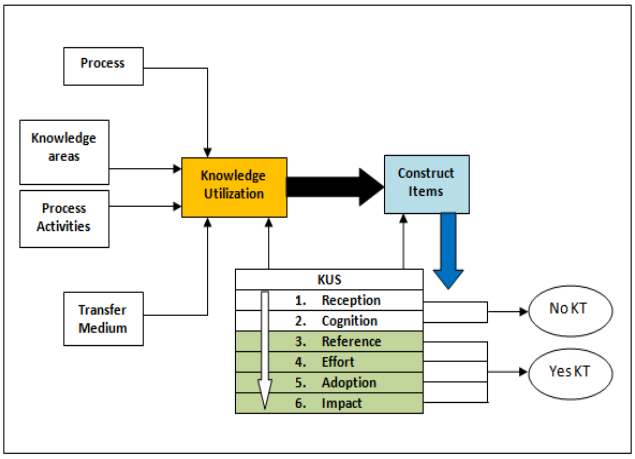

Figure 1.0 Knowledge Utilization Characterizing Model

V CASE STUDY

In this research, our interest lies in determining the occurrence of KT as well as to find the extent of knowledge utilization during software architecture development among both analyst and software architect teams. We interviewed 30 respondents from various expert positions including project manager, software architect and system analyst from different software houses and projects (Refer Table 2.0). All interviews were conducted in semistructured form according to the participants' own time and venue preferences. Each session took about 30 minutes. These respondents claimed to have been engaged in KT at their workplaces.

Table 2. Demographic Profiles for 30 Respondents

Resp.	Job title	Organization / location	Role held	KT under-standing	Engage-ment in KT
1	System analyst	Government/ Serdang	System Analyst	Good	Yes
2	System analyst	Government/ Serdang	System Analyst	Good	Yes
3	System Analyst	Government/ Serdang	System Analyst	Good	Yes
4	Project Manager	Government/ Serdang	Software Architect	Very Well	Yes
5	System analyst	Government/ Serdang	System Analyst	Good	Yes
6	System analyst	Government/ Serdang	Software Architect	Good	Yes
7	System Analyst	Government/ Serdang	System Analyst	Good	Yes
8	Project Manager	Government/ Serdang	Software Architect	Very Well	Yes
9	System analyst	Government/ Serdang	System Analyst	Good	Yes
10	System analyst	Government/ Serdang	System Analyst	Good	Yes
11	System Analyst	Government/ K.Lumpur	System Analyst	Very Well	Yes
12	System analyst	Government/ K.Lumpur	System Analyst	Good	Yes
13	System analyst	Private/ Sg. Besi	System Analyst	Good	Yes
14	Project Manager	Private/ Sg. Besi	Software Architect	Good	Yes
15	Project Manager	Government/ Serdang	Software Architect	Very Well	Yes
16	Project Manager	Private/ Cyberjaya	Software Architect	Good	Yes
17	System analyst	Private/ Cyberjaya	System Analyst	Good	Yes
18	System Analyst	Private/ Cyberjaya	System Analyst	Good	Yes
19	Project Manager	Private/ Cyberjaya	Software Architect	Very Well	Yes
20	System analyst	Government/ K.Lumpur	System Analyst	Good	Yes
21	Project Manager	Government/ K.Lumpur	System Analyst	Very Well	Yes
22	System Analyst	Private/ K.Lumpur	System Analyst	Good	Yes
23	Project Manager	Private/ K.Lumpur	Software Architect	Good	Yes
24	Software architect	Private/ K.Lumpur	Software Architect	Good	Yes
25	Software architect	Private/ K.Lumpur	Software Architect	Very Well	Yes
26	Software architect	Private/ K.Lumpur	Software Architect	Good	Yes
27	Project Manager	Private/ Cyberjaya	Software Architect	Very Well	Yes
28	Software Architect	Private/ PetalingJaya	Software Architect	Very Well	Yes
29	Software Architect	Private/ Petaling Jaya	Software Architect	Good	Yes
30	Project Manager	Private/ Petaling Jaya	Software Architect	Good	Yes

A list of 15 items were presented and used during the interview sessions. As anticipated, 100% of the participants agreed to have performed all of the listed items regarding knowledge utilization (Refer Table 2.1). This provides evidence that they have engaged in KT. This is also consistent with the requirement or prerequisite of successful KT that emphasizes putting the knowledge into action and not merely transferring and receiving knowledge.

Table 2.1 Detailed Result

Items	Frequency (and percentage %)		
	Somehow agree	Agree	Somehow agree
Using the knowledge gained from the mentoring session held prior to starting the project, we analyze software requirements.	0 (0%)	28 (93.3%)	2 (6.7%)
We held regular meetings and discussions for both teams in order to ensure we understand business and customer needs before development begins.	0 (0%)	26 (86.7%)	4 (13.3%)
We capture software specifications from business requirements described by the clients through brainstorming session.	0 (0%)	21 (70%)	9 (30%)
Using our architectural and design knowledge, we articulate and refine architectural requirements.	11 (36.7%)	18 (60%)	1 (3.3%)
Using our knowledge in software development methods, we document the defined requirements to produce Software Requirement Specification (SRS).	0 (0%)	27 (90%)	3 (10%)
Through several meetings and progress reviews, we get input on needs to evolve and improve the architecture.	0 (0%)	28 (93.3%)	2 (6.7%)
We create/draw the initial architecture based on an analysis of the given requirements.	3 (10%)	24 (80%)	3 (10%)
We often use reference architecture and make some adjustments to save time on architectural decisions.	10 (33.3%)	20 (66.7%)	0 (0%)

We make design decisions based on mutual agreement with the other team.	4 (13.3%)	18 (60%)	8 (26.7%)
Using our architectural and design knowledge, we identify the style and articulate the principles and key mechanisms of the architecture partitioning the system.	9 (30%)	16 (53.3%)	5 (16.7%)
We define how the various components fit together.	3 (10%)	27 (90%)	0 (0%)
We evaluate the architecture through various means including prototyping, reviews, and assessments.	5 (16.7%)	25 (83.3%)	0 (0%)
We do trade-off analysis on the design through active discussions with the business/software analyst team.	4 (13.3%)	24 (80%)	2 (6.7%)
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.	2 (6.7%)	22 (73.3%)	6 (20%)
We prepare architectural documents and deliver presentations to the stakeholders and other development teams.	0 (0%)	27 (90%)	3 (10%)

Finally, to determine the level of knowledge utilization from the specified activities in software architecture development, we mapped every questionnaire items with the six KUS (Refer Table 2.2 below).

Table 2.2 Mapping of Questionnaire Items With Stages of Knowledge Utilization

Items	Stages of Knowledge Utilization
1. Using the knowledge gained from the mentoring session held prior to starting the project, we analyze software requirements.	Effort
2. We held regular meetings and discussions for both teams in order to ensure we understand business and customer needs before development begins.	Effort
3. We capture software specifications from business requirements described by the clients through brainstorming session.	Reference
4. Using our architectural and design knowledge, we articulate and refine architectural requirements.	Effort
5. Using our knowledge in software development methods, we document the defined requirements to produce Software Requirement Specification (SRS).	Adoption
6. Through several meetings and progress reviews, we get input on needs to evolve and improve the architecture.	Adoption
7. We create/draw the initial architecture based on an analysis of the given requirements.	Adoption
8. We often use reference architecture and make some adjustments to save time on architectural decisions.	Implementation
9. We make design decisions based on mutual agreement with the other team.	Adoption
10. Using our architectural and design knowledge, we identify the style and articulate the principles and key mechanisms of the architecture partitioning the system.	Adoption
11. We define how the various components fit together.	Action
12. We evaluate the architecture through various means including prototyping, reviews, and assessments.	Adoption
13. We often do trade-off analysis on the design through active discussions with the business/software analyst team.	Implementation
14. 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.	Action
15. We prepare architectural documents and deliver presentations to the stakeholders and other development teams.	Impact

According to the model proposed, stage 3 onwards in knowledge utilization indicate the occurrence of uccessful KT. Hence, we can conclude that all respondents not only have proven their engagement in KT but also managed to maximize the benefits from KT. This result simply suggests that they really understood what is actually meant by KT.

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 architecting development does not only address the utilization of knowledge but put the emphasis in the essentials of learning from others and their experiences.

VI CONCLUSION

Software architecture development is a creative process that requires integration of different knowledge expertise and functionalities hence, open up wide opportunities for those involved to learn from each other. Nevertheless, many assertions have

been made about the flow of knowledge transfer being unclear and abstruse therefore led to failure in maximizing KT benefits. Based on the premise that knowledge is effectively transferred only when it has been put into use, we believe the right way to accomplish our intention is by characterizing how knowledge is utilized during the process. The utilization of knowldge or knowledge utilization is defined as the action of putting the knowledge received into use and translating it into useful and effective process, as well as project deliverables. We proposed a model based on PKAMI, and used it to verify the occurrence of successful KT as well as finding the extent of knowledge use. We believe it is an amazing effort that can assist software practitioners to better strategize on improving themselves and keep on producing quality software project deliverables. We are also aiming at elevating the essence of knowledge utilization to encourage those involved in development to find ways and opportunities to learn from others' experiences.

REFERENCES

Abou-Zeid (2008). *Knowledge management and business strategies: theoretical frameworks and empirical research*. IGI Global

Abdullah, S. (2012). *Knowledge Transfer Model of Team Capability in Non-Collocated Sofwtare Architecture Development: A Doctoral Thesis*. University Putra Malaysia, Malaysia.

Argote, L. (1999). Organizational Learning: Creating, Retaining, and Transferring Knowledge. Norwell, MA: Kluwer.

Argote, L., McEvily, B., and Reagans, R. (2003): "Managing Knowledge in Organizations: An Integrative Framework and Review of Emerging Themes", *Management Science*, 49(4), 571-582.

Darr, E. D., and Kurtzberg, T. R. (2000). An Investigation of Partner Similarity Dimensions on Knowledge Transf*er*. *Organizational Behavior and Human Decision Processes*, 82(1), 28-44.

Gardner, H. K., Gina, F., and Staats, B. R. (2011). Dynamically Integrating Knowledge in Teams: Transforming Resources into Performance. *Working Paper*.

Gonzalez, R. V. D., and Martins, M. F., (2017). Knowledge Management Process: a theoretical-conceptual research. *Gest. Prod.*, Sao Carlos, 24(2) 248-265.

Haas, M.R., and Hansen, M.T. (2005). When using knowledge can hurt performance: the value of organizational capabilities in a management consulting company. *Strategic Management Journal* 26(1), 1–24

Henninger, S., (2001), Keynote Address: Organizational Learning in Dynamic Domains, In: *Proceedings of the Learning Software Organization*, 8-16.

Knott, J., and Wildavsky, A. (1980). If Dissemination Is the Solution, What Is the Problem? Knowledge: Creation, Diffusion, Utilization, 1:4, pp. 537-578.

Paivarinta, T., and Smolander, K., (2015). Theorizing about software development practices. *Science of Computer Programming*, 124-135.

Polyaninova, T. (2011). Knowledge Management in a Project Environment: Organisational CT and Project Influences. *Vine*, 41(3).

Porrawatpreyakorn, N., Quirchmyer, G., and Chutimaskul, W. (2009). Requirements for a Knowledge Transfer Framework in the Field of Software Development Process Management for Executive Information Systems in the Telecommunications Industry. *IAIT 2009*, *CCIS* 55, 110–122.

Unphorn, H. and Dittrich, Y. (2010). Software architecture awareness in long term software product evaluation. *The Journal of Systems and Software*, 83.