International Journal of Sociologies and Anthropologies Science Reviews (IJSASR), 2 (5): September-October 2022, page 29-38, ISSN: 2774-0366
Website: https://so07.tci-thaijo.org/index.php/IJSASR/index
DOI: https://doi.org/10.14456/jsasr.2022.36

Impact of Organizational Factors and Individual Factors on Knowledge Usage

Yanee Nakprasom[1*], Guohua Chang[2], and Ao Chen[3]
[1,3]Panyapiwat Institute of Management, Thailand
[2]Rangsit University, Thailand
[1]*Corresponding author E-mail: yaneenak@pim.ac.th, ORCID ID: https://orcid.org/0000-0001-6655-1751
[2] E-mail: 510716549@qq.com, ORCID ID: https://orcid.org/0000-0003-2855-6580
[3]E-mail: freedom_cha@msn.com, ORCID ID: https://orcid.org/0000-0002-2167-5547

Received 02/09/2022
Revised 03/09/2022
Accepted 04/09/2022

Abstract: - *Both organizational environments and individual demands offered a multitude of factors that influenced the use of knowledge in the real working context. This paper demonstrates the factors at both organizational and individual levels not only affect programmers' knowledge choice and usage in the work but also mutually influence and interact with each other. For example, in order to cope with time pressure, some programmers were inclined to choose a conservative but quick and safe approach to complete the work, which is characterized by convergent thinking and behavior. Others tended to go for a more advanced and creative method which was labeled as trying something different on their knowledge usage. Moreover, habit and passion to use newly acquired knowledge on the job all represent the manifestation of individualism in knowledge usage. However, convergent thinking mirrors the expression of collectivism. Although they seem not to be theoretically matching each other, they do coexist in practical work.*

Keywords: Knowledge Usage; Organizational and Individual Factors; Timing; Convergent Thinking; Habits; Individual Technique; Passion to Use Newly Acquired Knowledge

Introduction

Effectively managing the use of knowledge in work has always been seen as a challenge. On the one hand, perhaps this is because knowledge usage in most cases was closely connected with its owners' behavior and consciousness. Individual differences brought huge uncertainty in the process of knowledge application. On the other hand, the organizational environment and factors also affected how knowledge was used in particular contexts. This paper identified a number of factors at both organizational and individual levels that influenced knowledge usage in programmers' work. Some of these factors were also reflected in existing knowledge management literature when it discusses transferring knowledge and learning into practice.

Objective and Scope

This paper demonstrates the factors at both organizational and individual levels not only affect programmers' knowledge choice and usage in the work but also mutually influence and interact with each other.

Literature Review

The organizational environments offered a multitude of factors that influenced the use of knowledge in programmers' work. At the organizational level, these factors included time pressure and the phenomenon of convergent thinking.

Organizational level factors: Timing issue affecting knowledge choice and usage

Time was one of the most important factors affecting what knowledge was chosen to undertake the job and finish it. Time pressure not only affected knowledge usage but also conveyed a sense that too much work needed to be done in a set time. Thus, time pressure was always accompanied by an excessive workload according to programmers. The choice of method and knowledge usage was considered very crucial. How to quickly and safely complete tasks while continuing to develop capability on the innovation of skill and knowledge in work has become the main issue programmers needed to consider and the challenge they needed to face. However, often in practical work, you could not have the cake and ate it. According to different work situations, programmers had to choose to privilege one over another. For example, at the beginning of a software development project, as time was still plenty, no one would feel too much pressure on their work. As a result, time was not usually used very effectively by programmers compared with the later period of a project. Many members of a team might consciously or unconsciously spend too much time on some components of work to make it good so that they unconsciously did not leave enough time for the rest of the work. In the later period of the project, everybody more or less had a lot of work left that needed to be done. Especially, when management kept pushing programmers' work progress, they would feel more pressure on both time and workload. In order to hit the deadline, they might turn to use the fastest way to finish the rest of their work. As a result, sometimes the result of their work was even worse than the standard level. After all, sticking to the deadline to finish the work was the priority of their job. Using new methods could be seen as an extra bonus. If there was an innovative attempt in the work, it was good. If there was not, that was fine as well. Especially, using new methods might not always lead to a good result. Sometimes it might cause trouble as well in the operation and integration of the product.

However, some programmers argue that in the work, spending some time exploring new methods to make their work better could not be simply treated as wasting time or risky behavior. After all, trying to develop capability on innovative methods to solve the problem in the work was also a channel to cope with enormous time pressure and increase performance as well as to be beneficial for the improvement of both organizational and individual knowledge.

Thus, trying new methods in their work could also be considered as another way to cope with time pressure through improving unit performance. However, in order to deal with time pressure and finish work quickly and safely, some programmers tended to pursue and imitate the ways that other people used so as to minimize the opportunity of making mistakes.

Organizational level factors: Convergent thinking on the technical routes and methodologies affecting knowledge usage

Convergent thinking was identified as another important factor affecting the use and reuse of knowledge in these programmers' work. Convergent thinking means that people tended to use what other people did on the job as the norm for their own behavior and operating method. In other words, many programmers tended to consciously observe, learned, and imitate what other people did and then reuse it in their own work.

To elaborate, shorter time to market, better quality and more effective productivity has become the goals that every software development company pursues according to these respondents at least. Thus, how to more effectively use and reuse knowledge was seen as a very important tactic to improve productivity and effectiveness as it could prevent the repetition of past mistakes and accelerate work progress. This belief was apparently not only accepted by management but also deeply influenced these programmers' thinking and behavior. As a result, in order to avoid making mistakes, many programmers, especially newcomers and less experienced programmers, were more likely to imitate exactly what other people did on the job rather than find out their own ways. Especially when a series of ready-made ways existed for doing certain kinds of work, programmers were more likely to use these technical methods instead of searching for anything better. Further, using other people's methods was considered a usable way to avoid losing face in front of one's boss by making too many mistakes. After all, if a programmer used the traditional methods which everybody used, they would not be marked as a worker without ambition. But if a programmer always created problems in their work due to trying something new, they might lose their job. Moreover, the advantage of convergent thinking was that it promoted knowledge transfer from senior programmers to junior programmers and unconsciously helped to form technical traditions and common-sense knowledge ground in the organization.

However, even though these technical routes and methodologies were well known in the company or the project team and had been seen as the traditional and accepted way to complete the work, there was no convincing evidence to prove that these ways were the best or the most effective methods. Especially for knowledge and methods gained from informal learning, they were probably used by previous programmers and then inherited by their colleagues and finally formed a technical tradition, which certainly lacks of reliable test about their effectiveness. Along with the rapid development of technology and continual enhancement of work complexity, the applicability of this technical tradition, especially formed by informal ways, was also challenged by some programmers in their work.

Thus, although convergent thinking on knowledge usage had been identified as an obvious behavioral characteristic in programmers' work, inconsistent behaviors still took place occasionally. Some programmers did not follow these traditions one hundred percent. From time to time, they also tried to use new methods or knowledge to deal with problems in their work. Nevertheless, due to new creative knowledge behaviors might not always bring positive results or enhancement of effectiveness, these methods had not been widely disseminated and accepted in the organization. The next section picks up factors at the individual level that impacted knowledge choice and usage in programmers' work. According to my data, habit, individual technique and relevant knowledge, and passion to use newly acquired knowledge were identified by programmers as the most influential factors which affected their knowledge choice and usage.

Individual level factors: Habit affecting knowledge choice and usage

Although programming work was well known as a very objective and standardized job that included a lot of usage of systematic knowledge, it was also inevitably influenced by many subjective factors. First of all, one of the main subjective factors influencing knowledge choice and application within these programmers' work was the individual's technical habit. When a person initially used certain knowledge or technique, this person was most likely to engage in active cognitive processing. Subsequently, along with continually repeated behavior, this cognitive processing gradually dissipated over time, instead, routinized behavior steadily formed and finally become a 'habit' (Jasperson *et al.*, 2005).

In previous research, there were only a few works that linked individual technical habits to knowledge choice and usage. However, although knowledge choice and usage tended to be seen as behaviors with a conscious intention, it is important to be aware that some or much knowledge usage within programmers' work might be habitual. To elaborate, software development projects according to my data differed substantially from each other, and similar products could be achieved by different methods and using different programming knowledge. This really depended on individual programmers' habits and choices sometimes. Although software development knowledge was generally seen as objective and standardized, there was still a certain amount of room or we could say that programmers still could find a way to exert their subjective competence and influence how the work was done. Therefore, according to these findings, it was reflected that some programmers preferred to use, for example, the technical method A, but others might get used to using the technical method B or the technical method C.

Nevertheless, not being wrong did not represent that it was good. Sometimes different technical habits between programmers could affect the quality of work. Moreover, the habit was more like a kind of unconscious reflex that was repeated regularly when people undertake routine tasks. It was first established by intentional behavior. But when the use of a particular technique, method, or knowledge became habitual, less cognitive planning is involved (Cheung and Limayen, 2005). It becomes imprinted in these programmers' neural pathways and very difficult to break. Moreover, programmers reflected that there was a lack of attention to knowledge usage in their work process. Nobody actually cared how knowledge was used unless something went wrong in the work.

Individual level factors: individual technique and relevant knowledge affecting knowledge choice and usage

The technical knowledge and skills of a person sometimes influenced what method they used and which technical direction they liked to go in. In other words, what people knew decided what they could do and what they were able to think. In software development knowledge, there are many different types of techniques and programming languages. Within these programming languages and techniques, there are also many different sub-technical routes. An experienced senior programmer might be an expert in one or two of these technical areas but would not be good at all of them. Therefore, if someone was good at a certain technique, he/she was likely to keep working on this technical area and to continue to develop it into a higher level and make themselves more expert in this area.

Moreover, when one aspect of a person's knowledge was continually improved, other aspects of their knowledge would relatively slow down. Thus, when someone was recognized as an expert in an area, he/she was more likely to be allocated to work on what he/she was good at in the work rather than other aspects of knowledge or skill that he/she was not as perfect in.

In addition, the technical knowledge of a person not only affected what these programmers could do in their work but also influenced what they thought when met problems. In other words, people tended to work out things from what they already knew. It was very rarely that people developed brilliant ideas or effective solutions which were out of their knowledge circle. It was much easier to be creative or do things smart in the technical field which they were already good at. Therefore, the technical knowledge of a person not only decided what this person could do but also to a certain extent affected what this person could think and suggest.

Individual level factors: Passion to use newly acquired knowledge on the job affecting knowledge usage

These programmers sometimes wished to practice newly acquired techniques or knowledge in their current work that they learned either formally or informally. For programmers, trying something new on the job was exciting, a good opportunity to familiarize, practice, and quickly master what they had just learned, and a great chance to improve their work-related knowledge. Even though sometimes it was not an appropriate situation in which to use these newly acquired knowledge or techniques, programmers still wanted as possible as they could to try them out on their job.

These responses above seemed to indicate that these programmers saw the application of newly acquired methods or knowledge to real work as a wonderful opportunity to practice and improve their skills. They appreciated being given chances to experiment with what they learned on the job. However, this was not always the case. Some respondents had different ideas. They did not completely support the idea of letting programmers freely try out newly acquired knowledge on the job, especially when the result of the attempt would influence other people's work. Some programmers argued that many newly acquired tricks either had not been mastered very well by programmers or the trick itself did not suit the actual demands of the situation. Thus, without any careful thought using newly acquired knowledge on the job might cause unnecessary problems in the project and waste everybody's time. To elaborate, software development was a very complicated process that often needed many people working together to make one product work properly. Thus, collaboration and making sure each person's work was compatible with others in a big project were very important. Any tiny detail that went wrong might cause a huge functional problem. Keeping safe and stabilized working methods and reducing unnecessary errors in the coding stage was very important. Thus, trying not to apply unfamiliar and unsure techniques in the project was seen as one of the ways to ensure the effectiveness of the work. These responses reflected another side of consideration that using newly acquired knowledge on the job might not always result in benefits. It caused trouble too sometimes. However, these ideas were fairly isolated accounts. In general, the majority of responses still demonstrated a positive attitude about using newly acquired knowledge on the job.

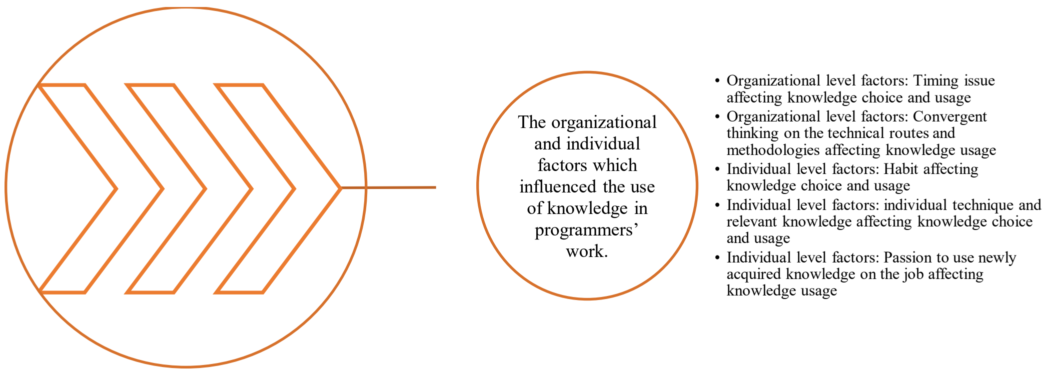
The organizational and individual factors which influenced the use of knowledge in programmers' work.
• Organizational level factors: Timing issue affecting knowledge choice and usage
• Organizational level factors: Convergent thinking on the technical routes and methodologies affecting knowledge usage
• Individual level factors: Habit affecting knowledge choice and usage
• Individual level factors: individual technique and relevant knowledge affecting knowledge choice and usage
• Individual level factors: Passion to use newly acquired knowledge on the job affecting knowledge usage

Figure 1: The organizational and individual factors which influenced the use of knowledge in programmers' work.

To elaborate, two groups between Organizational factors and Individual factors, it can be summarized as the following table.

Table 1 Organizational and individual factors affecting the knowledge usage

Organizational factors	**Individual factors**
-Timing issue	-Passion to use newly knowledge
-Convergent thinking on the technical routes and methodologies	-Individual technique and relevant knowledge
	-Habits

Research Methods

This paper adopts qualitative research approach. Instead of investigate any statistic result, this paper is going to explore the in-depth data from the targets to reflect the social reality. A case analysis approach was conducted with programmers. The semi-structured interviews were adopted to collect the qualitative data and thematic analysis was used to analyse the data.

Discussion

The above findings have explored different factors at both organizational and individual levels that influenced knowledge choice and usage within programmers' daily work. At the organizational level, we can see that two main factors influenced knowledge usage in these programmers' work: timing issues and the phenomenon of convergent thinking. Neither has been thoughtfully discussed in the existing knowledge management literature relating to knowledge usage within programmers' work. The timing issue, reflected that time pressure somehow affected the way people did things. If under such pressure, many programmers might choose the fastest or the most familiar method to get their job done rather than attempting to explore any new method to make things better. As Keegan and Turner (2001) highlight in their study, the exploitation of knowledge will not occur as anticipated if it is under time pressure. Although we cannot simply assert that the method programmers used when they were under time pressure was definitely lower quality, what we could say was that they could make it better if they had enough time according to my findings at least. Moreover, the issue of time pressure was not only caused by the tight time schedule of the management and the client but also was perhaps created by the poor time management skills of programmers. As the findings above reflected, perhaps if programmers could do things quicker in the early stage, they might not feel so tight in the end. However, it was really difficult to tell which was the most significant pressure – did management try to push too hard on project time, or was it that programmers did not manage good enough time in their work? But what we can tell from the findings is that a tight time schedule led to pressure on programmers. Such pressure made programmers have to make a choice and balance the use of their knowledge. On the one hand, programmers needed to complete work on time and avoid making any mistakes. On the other hand, programmers wanted to reduce time pressure and improve their knowledge through attempting to develop new and more effective methods to deal with problems in their work. Originally, these two purposes were not contradictory behaviors. However, due to time pressure, they eventually become the dilemma that programmers had to face in their work.

For the phenomenon of convergent thinking, many programmers tended to have a convergent attitude as to how to do certain work. They preferred to learn methods from others and to do exactly what they did in the work so as to avoid making mistakes. Ibarra (1999) points out that individuals develop practices by observing and imitating others at the beginning of their work to form their own practices which match the wider organization's norms. Similarly, Amin and Roberts (2008) indicate that learners learn to master the community's knowledge by beginning with observing, imitating, and practicing exactly what other people do. The advantages of this convergent thinking phenomenon were that traditional technical routes and methodologies could be passed from seniors to juniors, common technical knowledge was formed and kept inside the organization, and the repetition of past mistakes was to a certain extent avoided. Individual programmers can also quickly adapt to a new work environment and achieve the work at an acceptable level.

However, as convergent thinking and activities are primarily concerned with learning and replicating existing knowledge rather than engaging in radical innovation, it can make existing knowledge very powerful and dominant while leading to the stickiness and rigidity of this knowledge in the organization which may block the development of new knowledge (Levinthal and March, 1993; Bettis and Prahalad, 1995; Brown and Paul, 1998; Crossan *et al.,* 1999). For example, the traditional routes and methodologies might become the only route and methodology in the organization and the opportunity for the use of other knowledge or methods could be reduced in practice. Programmers easily formed and used the so-called standardized and collective methods and might consciously or unconsciously avoid the emergence of 'doing things differently. As Roberts (2006) states, knowledge that is aligned with the particular predispositions of a community and supports their current working practices is likely to be adopted and expanded in the organization than the knowledge that challenges current practices. Similarly, Nelson and Winter (1982) point out that ways of doing things can become very patterned in the organization which is very difficult to be changed or replace. Thus, while being aware of the effect of convergent thinking and activities on avoiding making mistakes and facilitating newcomers' work in their departure, it is also worth realizing the possible obstruction it created for the creation and use of new knowledge.

At the individual level, the findings mirrored three different factors which influenced knowledge usage in these programmers' work: habit, individual technique and knowledge, and the passion to use newly acquired knowledge on the job. Habit and individual techniques and knowledge have not been discussed in the existing knowledge management literature. But the factor of the passion to use newly acquired knowledge on the job can be considered as closely linked to the concept of 'the opportunity to use the knowledge which has been mentioned in the relevant literature (e.g., Leo, 2001; Lim and Johnson, 2002). Therefore, my findings of passion to use newly acquired knowledge on the job empirically confirmed the claim of some existing research and push it forward to concentrating on the area of programmers' jobs. These data reflected that many programmers liked to use newly acquired knowledge on the job and to try to play with it even if sometimes this newly acquired knowledge did not quite fit into the situation. As Newell *et al.* (2009) indicate, since projects stand outside traditional hierarchical controls, allowing those involved to have more discretion and autonomy to 'do things differently. Through absorbing new knowledge and putting them into their actual task, programmers could familiarize themselves with this knowledge, quickly master these new skills and build up their effectiveness. As Lim and Johnson (2002) suggested, having an opportunity to use what is learned on the job is a very important channel that could transfer the learning from training programs to practical work. However, there is a very fine line between the adapted use of newly acquired knowledge and inadequate use of newly acquired knowledge in the work which can potentially distract from performance and lead to trouble. Especially the newly acquired knowledge did not fit into the actual situation of work. In the case of my study, some programmers revealed that the immoderate use of newly acquired knowledge on the job caused problems in the operation of the software product. As Newell *et al.* (2009) argue, no organization of any scale can survive for long if every situation is reared as an opportunity to learn and experiment to do things differently. However, there was no agreement on whether and to what extent newly acquired knowledge should or should not be used in their everyday work practice. A certain number of respondents who supported the use of newly acquired knowledge on the job was to consider this issue from the point of view of how to quickly improve their own knowledge and skills so as to better adapt to their work. Some others argued against it was to think over this issue from the angle of how to ensure to complete their daily task safely and smoothly. None of their consideration can be simply evaluated as wrong in practical work. They are just two different working philosophies held by different programmers and applied to guide knowledge usage in their everyday working practice.

Habit is another influential factor in knowledge usage which is reflected in programmers' work. Butler and Hope (1995) explain habit as a well-learned action sequence of a person which was developed through a long period of learning, imitating, and practicing in a stable context. It could be repeated without conscious intention when triggered by environmental cues (Guinea and Markus, 2009). Bourdieu (1977) describes habit as a system of disposition that is acquired from the historical past and turns into one's nature and predominates one's behavior without sensing it. It can be an individual unique reflection of a specific situation marked by personal characteristics. According to Tsoukas (1996), knowledge usage should be considered as an individual case rather than a uniform case due to individuals carry the knowledge and respond to situations uniquely. Or it can also be a reflection of a part of group working style as the individual who belongs to a social grouping may be more or less influenced by their group's values, behavior standard or working style which forms their working habit. As Bourdieu (1990) implies, the members of the same group are more likely to exhibit a similar habit that developed from the commonalities of their experience. In the case of knowledge usage in programmers' work, although the whole working procedure was directed by management, the specific details of treatment: what and how knowledge was applied, still depended on individual programmers a lot and might be different from one situation to another. This difference was to a certain extent influenced by personal habits on knowledge usage in the work. Many programmers indicated that they had their own habitual way to perform their techniques or skills on the job. How to use knowledge really depended on their personal habits which stemmed from experience being accumulated in practice. For example, some programmers got used to use some complicated coding formulas which were very difficult to make them to change in spite of their way might not be very efficient. The force of habit was very powerful, sticky, and difficult to change. Even if someone's habit can be temporarily changed due to others' reminding or monitoring, he/she might unconsciously change back later. As Bourdieu (1977) describes, individuals make decisions based on prior experiences but in situations where no such experience exists, such individuals would still be prone to draw on similar previous experiences to determine a course of action. Unless there is a breakdown to interrupt our habitual and comfortable working behavior which leads to reconsidering our fundamental thinking and habit, otherwise the existing working habit is hard to vary (Nonaka and Takeuchi, 1995).

Data about individual techniques and relevant knowledge suggested that the technique and knowledge of a programmer reflected what this programmer could do and what this programmer was able to think. If a programmer was good at certain technique, it was likely that on the one hand, they were more like to use their technical forte to complete their work or resolve relevant problems in the work, on the other hand, they would be frequently allocated to work on this technical type of project so that they could maximize to exert their technical forte to increase performance and productivity. As a result, programmers' certain techniques continued to grow and be made perfect while others were deserted. Moreover, individual technique strength not only affected what kind of work they could do but also influenced the angle from which they looked at the technical problem and how they worked out the solution. In these findings, programmers were more likely to start to think about a solution for a problem based on what they already knew in their technical area. It was rarely that programmers started to consider the solution for a problem from some other technical areas which they were not as familiar with. Even if sometimes the task was not suited to be conducted by the technique that the programmer was good at, he/she would still like to design and find out a way to use his/her technical forte to cope with the work.

Conclusion

This paper reflected that two main factors at the organizational level affected knowledge application in programmers' daily work: time and convergent thinking. The timing issue revealed that there is no any obvious knowledge application pattern in programmers' work. Depending on their different situations, they chose different ways to implement their work in order to cope with time pressure and heavy workload in their actual work. According to how much the remaining time left in their work, they also adjusted their method or applied knowledge in the process of the task to guarantee completing the work on time. Thus, although time pressure had been identified as one of the most important factors influencing knowledge choice and usage in programmers' work, there is no obvious agreement of how knowledge should be used in their work to cope with time pressure. Moreover, the phenomenon of convergent thinking pointed out that in work programmers preferred to a relatively conservative and safe way to doing things which is characterized by learning and imitating other people's method and using it into their own task so as to avoid making mistakes as well as guarantee an acceptable result.

This paper also suggested that three main factors found at the individual level affect the knowledge usage in programmers' work: habit, individual technique, and relevant knowledge and passion to use newly acquired knowledge on the job. The first considers that different habits to get the job done led to different ways that knowledge was used in the work. Most of these habits were unconscious and habitual behavior so they were very difficult to be changed and deeply embedded in their personal behavior system and reflected on their knowledge choice and usage. The second reflects that the technique and knowledge of a programmer might to a certain extent decide what this programmer could do and what the programmer was able to think and suggest when he/she met a problem in the work. On the one hand, programmers were inclined to initiatively use what they were good at to deal with problems in the work. On the other hand, programmers were often passively allocated a lot of tasks located in their specific technical forte. As a result, one aspect of knowledge was well developed while improvement of other aspects of knowledge was to a great extent obstructed. The last explains that programmers were keen to attempt to use newly acquired knowledge in their work. On the one side, it was seen as a good opportunity for programmers to familiarize themselves, practice, and quickly master the knowledge just learned. On the other side, it was also considered as a risky behavior that might lead to trouble or problem to programmers' work.

Recommendations

The important implication of this paper was that while many factors at both organizational and individual levels influence how knowledge is used in programmers' work, these factors are also interrelated, interacted and affected with each other, which even makes the situation more complicated. There is no obvious knowledge application pattern that can be followed. Programmers' work is actually an outcome of that the use of knowledge continually being influenced, being adjusted, and being compromised in the process of work so as to gain an acceptable level.

References

Amin, A., & Roberts, J. (2008). Knowing in Action: Beyond communities of Practice. *Research Policy*, 37 (2), 353-69.

Bettis, R., & Prahalad, C. K. (1995). The Dominant Logic: Retrospective and Extension. *Strategic Management Journal,* 16(1), 5-14.

Bourdieu, P. (1977*). Outline of a Theory of Practice*, trans. R. Nice. Cambridge: Cambridge University Press.

Bourdieu, P. (1990). *The Logic of Practice.* Cambridge: Polity Press.

Brown, J.S., & Duguid, P. (1998). 'Organizing Knowledge'. *California Management Review,* 40(3), 90-111.

Bulter, G., & Hope, T. (1995). *Managing Your Mind: The Mental Fitness Guide.* Oxford: Oxford University Press.

Cheung, C. M. K., & Limayen, M. (2005). The Role of Habit in Information Systems Continuance: Examining the Evolving Relationship Between Intention and Usage. *Proceedings of the 26th International Conference on Information Systems*, D. Avison, D. Galletta, and J. I. DeGross (eds.), Las Vegas, NV, December 11-14, 471-482.

Crossan, M.M., Lane, H.W., & White, R.E. (1999). An Organizational Learning Framework: From Intuition to Institution. *Academy of Management Review,* 24(3), 522-537.

Guinea, A.O., & Markus, M.L. (2009). Why Break the Habit of a Lifetime? Rethinking the Roles of Intention, Habit, and Emotion in Continuing Information Technology Use. *MIS Quarterly*, 33(3), 433-444.

Ibarra, H. (1999). Provisional Selves: Experimenting with Image and Identity in Professional Adaptation. *Administrative Science Quarterly*, 44 (4), 764-791.

Jasperson, J., Carter, P.E., & Zmud, R.W. (2005). A Comprehensive Conceptualization of Post Adoptive Behaviours Associated with Information Technology Enabled Work Systems.' *MIS Quarterly,* 29(3), 525-558.

Keegan, A., & Turner, R. (2001). Quantity Versus Quality in Project-Based Learning Practices.' *Management Learning,* 32(1), 77-98.

Leo, S. (2001). *The Management of Transferable Skills: An Investigation into Manager's Perceptions and Practice of Transferable Skills Within the Context of Business Enterprises in South West England*, Unpublished Paper, Exeter University, Exeter.

Levinthal, D.A., & March, J.G. (1993). The Myopia of Learning.' *Strategic Management Journal,* 14(Special Issue), 95-112.

Lim, D.H., & Johnson, S.D. (2002). Trainee Perceptions of Factors That Influence Learning Transfer.' *International Journal of Training and Development*, 6(1), 36-48.

Nelson, R., & Winter, S. (1982). *An Evolutionary Theory of Organisational Change.* Cambridge: Harvard University Press.

Newell, S., Robertson, M., Scarbrough, H., & Swan, J. (2009). *Managing Knowledge Work*. Palgrave, Basingstoke, Hampshire.

Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation.* New York: Oxford University Press.

Roberts, J. (2006). Limits To Communities of Practice'. *Journal of Management Studies,* 43(3), 623-639.

Tsoukas, H. (1996). The Firm as A Distributed Knowledge System: A Constructionist Approach. *Strategic Management Journal,* 17(Winter Special Issue), 11-25.