European Journal of Computer Science and Information Technology
Vol.5, No.3, pp.14-27, June 2017
Published by European Centre for Research Training and Development UK (www.eajournals.org)

KNOWLEDGE MANAGEMENT AS A TOOL FOR MITIGATING SOFTWARE CRISIS

Bolanle F. Oladejo and Sanjo Ojutalayo

Department of Computer Science, University of Ibadan, Ibadan, Nigeria.

ABSTRACT: *Despite the enormous achievements in the field of Software Engineering (SE), in which software engineers had come up with methods, techniques, models, and tools to mitigate software crisis, yet many software projects fail to meet customers' needs with respect to delivered products, to meet schedule, and cost above budget. Thus, this paper designed a framework for the application of Knowledge Management (KM) such that knowledge of SE practitioners could be preserved and reused. An Experience Repository (ER) was designed and implemented. The design employed architectural and conceptual representation of ER for SE. ER explored Knowledge codification and Document Management techniques for knowledge acquisition. It also provided a motivational method for reinforcing KM culture among SE practitioners through the platform of a dashboard which provides a rating mechanism in order to encourage practitioners to make use of the system. The system provided a platform for retrieval and reuse of software developers' experience and projects' processes, artifacts and documentation. The application of the KM initiative such as the ER in this work serves as a tool to mitigate software crisis in software development projects as lessons from previous projects guide SE practitioners to avoid mistakes and to reuse knowledge from successful projects.*

KEYWORDS: Knowledge Management, Software Engineering, Software Crisis, Software Process Improvement (SPI). Experience Repository

INTRODUCTION

Software Engineering (SE) is the application of engineering to the design, development, implementation, testing and maintenance of software in a systematic method. (Abran, Alain; Moore, James.; Bourque, Pierre; Dupuis, Robert; Tripp, Leonard,2004). It is an engineering discipline whose focus is the cost effective development of high-quality software systems. "Software is abstract and intangible" (Somerville, 2009). It is not constrained by materials, neither is it governed by physical laws nor by manufacturing process. Somehow, this simplifies software engineering as there are no physical limitations on the potential of software. However, this lack of natural constraints can result into software that is extremely complex and hence very difficult to understand (Somerville, 2007). This complexity has eroded the enormous achievements in the field of software engineering, in which the software engineers had come up with tools, methods, models, techniques and others to mitigate software crisis, yet there are many software projects that fail to meet schedule, cost above budget, and deliver products that do not meet customer needs. (Standish Group Chaos report, 2015).

According to Somerville: *"While I was writing the 7th edition, a government enquiry in UK reported on the project to provide a national system to be used in courts that try relatively minor offenders. The cost of this system was estimated at £156 million and it was scheduled*

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

for delivery in 2001. In 2004, cost had escalated to £390 million and it was still not fully operational. There is therefore, still a pressing need for software engineering education" (Somerville, 2007, pp. viii).

In view of this, it is imperative for researchers and software industries to seek methods to mitigate this ugly trend. In software development, different approaches have been proposed to reduce project costs, shorten schedules, and increase quality. These approaches address factors such as process improvement, introducing new technologies, and improving practitioners' performance ("people ware"). Knowledge Management (KM) addresses mainly people ware.

Davenport and Thomas (1994) define KM as a process of capturing, developing, sharing and effectively using organizational knowledge. These qualities of KM make it relevant to mitigate the problem of software crisis. Because software development is a human and knowledge-intensive creative activity, KM acknowledges the importance of individuals having access to the correct information and knowledge when they need to complete a task or make a decision (Sven and Mikael, 2003).

For any organization to be competitive, such organizations must identity and effectively manage, protect and exploit its assets. In the case of Software Engineering (SE), Software Organizations' main assets are not plants, buildings, or expensive machines, but its intellectual capital (that is, knowledge of its main actors). This has to be efficiently managed and properly protected. Due to the fact that a large amount of organizational knowledge is not documented, many organizations lose knowledge when employees leave their organization. This increasing mobility of workers has created a growing need to retain employees and, more importantly their knowledge, which has led to a growing call for Knowledge Management (KM) (Sven and Mikael, 2003).

In the light of the import of KM to organizations, this work aimed at providing a framework for the application of KM initiative to software projects development to mitigate software crisis to the intent that knowledge of SE practitioners could be preserved and reused. We proposed a "Experience Repository" (ER) named "Imo-hub" (knowledge hub). The rest of this paper entails theoretical background of this study in section 2; the research methodology and development of KM system for Experience Repository in section 3 and 4 respectively. The paper is concluded in section 5.

Knowledge Management

Before we can define knowledge Management (KM), we like to establish a definition of knowledge, information and data, as well as establishing the difference between these concepts.

There is no single universally acceptable definition for knowledge as different scholars define it in different ways. According to Pohl, (2002) knowledge is defined as the addition of context to information, on the other hand, information is defined as the existing relationships among data, that is, numbers and words. Oladejo, Odumuyiwa and David (2010) define Knowledge as the perception an individual has about a fact or event in certain context. They perceived knowledge as "know-how". Knowledge is classified into two types, namely, the explicit (objective) and the tacit (subjective) knowledge. Explicit knowledge is the articulated knowledge in form of documents, operation manual, video, and so on. This type of

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

knowledge could be transmitted across individuals, formally and systematically. While tacit knowledge refers to the "know-how" of an individual. This is illustrated in figure 1

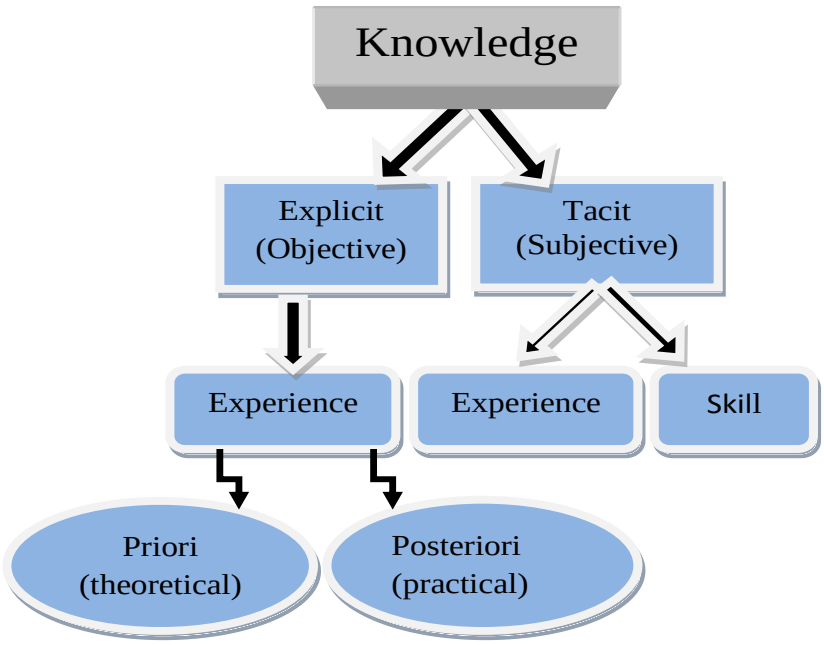

Figure 1: Classification of Knowledge (Source: Oladejo et al., 2010)

KM is an emerging discipline that promises to capitalize on organizations' intellectual capital. The concept of taming knowledge and putting it to work is not new; phrases containing the word *knowledge*, such as *knowledge based* and *knowledge engineering*, existed before KM became popularized. The artificial intelligence community has long dealt with knowledge representation, storage, and application. Software engineers have engaged in KM-related activities aimed at learning, capturing, and reusing experience, even though they were not using the term- "Knowledge Management." KM is unique because it focuses on the individual as an expert and as the bearer of important knowledge that he or she can systematically share with an organization. KM supports not only the knowhow of a company, but also, know-where, know-who, know-what, know-when, and know-why. (Ioana and Mikael, 2002).

Davenport (1994), defines Knowledge Management as the process of capturing, distributing, and effectively using knowledge. Michael (2012), describe Knowledge Management as a discipline that promotes an integrated approach to identifying, capturing, evaluating, retrieving and sharing all of an enterprise information assets.

As much as we want to agree with both Davenport and Michaels' definitions, we posited that KM does not stop at the utilization of knowledge but also extends to preservation for subsequent usage. We therefore redefine Knowledge Management as the process of effectively capturing, storing, distributing, utilizing and preserving the intellectual properties of an organization.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

The main focus of KM is to capture and make available, the information and knowledge that is in the people's head as it were and that required being explicitly set down, so it could be used by others in the organization.

Every organization is expected to gain knowledge from the daily work it does over the years. There are questions raised, such as, can this knowledge be captured? Is it manageable? How is knowledge different from information and data? Specifically for this research, we are looking at how software improvement can be achieved using KM approach. Hence, KM Initiative is an approach to improve productivity and software quality. KM initiatives, if effectively implemented will systematically and continuously improve the software producing organization's ability to produce and deliver quality software within time and budget constraints. Mathiassen, Pries-Heje, and Ngewenyama, (2001) argued that software process improvement efforts depend on the implicit, individual knowledge of practitioners in an organization. A major challenge for software process improvement initiatives is hence to integrate strategies and mechanisms for creation and utilization of knowledge for quality software development.

The application of KM techniques will prevent the software firms from re – inventing the wheel, an instance of this is making use of Experience Repository (ER) that is produced from this work. Consequently, it will bring about an improvement to the software development process. Software development is a quickly changing and knowledge intensive business. It usually involves many people working on different phases of the project. However, as the need for improvement is increasing, the available resources are not increasing along; yet software organizations require rise in productivity. Hence, this work focuses on application of KM to SE with the intent of bridging this gap in order to actualize capture, preservation and eventual use of software projects knowledge.

Software Engineering

Software engineering (SE) encompasses a process, a collection of methods (practices) and an array of tools that allow professionals to build high quality computer software (Pressman, 2010). SE makes it possible to build complex systems within a reasonable time frame and with high quality in response to the software crisis that predate the software engineering projects.

The name, Software Engineering was proposed in 1969 at a NATO Conference to discuss software development problems. Large software systems were late, did not deliver the functionality needed by their users, cost more than expected, and were unreliable (Sommerville, 2009). The term software engineering was chosen deliberately for the conference in order to be provocative with the negative trends becoming apparent, the conference felt that, it was necessary for software development to be performed with the rigor and discipline associated with other branches of engineering (Edwards,2003).

Software engineering is about teams. The problems to solve are so complex or large, that a single developer cannot solve them anymore. Software engineering is also about communication. Software Engineering teams usually does not consist of only developers, but also of testers, architects, system engineers, customer, project managers etc. Software projects usually require careful planning of their sizes. It involves more than just coding, but also following guidelines, writing documentation and also the unit testing. However, the units testing alone are not enough. The different pieces have to fit together (unit integration). And

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

we have to be able to spot problematic areas using metrics. SE activities does not end at coding, it has to be deployed and maintained. For large projects maintaining software can keep many people busy for a long time. Since there are many factors influencing the success or failure of a project; software engineer needs to learn a little of project management and its pitfalls, but especially what makes projects successful. Software Engineering involves the use of many tools just like any other field of engineering. In summary, SE involves series of phases such as requirements gathering, system design, implementation, testing, deployment and maintenance. This is usually called the Software Development Life Cycle (SDLC).

Software Crisis

The difficulty of writing code for a computer program which is correct, understandable, delivered as scheduled and cost within budget, with customer satisfaction is referred to as software crisis (Kartik, Lokesh and Kislay, 2014). The term software crisis was given by F.L. Bauer at the first NATO Software Engineering Conference in 1968 at Garmisch, Germany. The fundamental causes of software crisis revolves around three increasing demand, increasing complexity and increasing challenges with a corresponding increase in the workforce, methods and tools used.

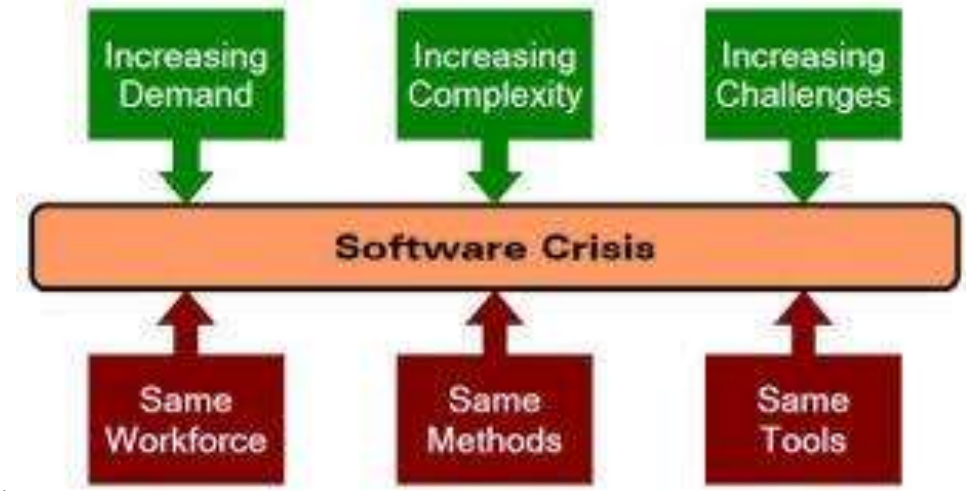

Figure 2: Illustration of Software Crisis Source: (Kartik et al, 2014)

Statistical Facts about Software Crisis

According to Standish Group's 2015 Chaos report, only one in three software projects will turn out to be successful. Sixty – six (66%) of technology projects (based on the analysis of 50,000 projects worldwide) end in partial or total failure. More surprisingly, these statistics have been the same for the last five years as evident in the report. Furthermore, seventeen (17%) of large IT projects go so badly that they can threaten the very existence of a company.

On average, large Software projects run 45% over budget and 7% over time, while delivering 56% less value than predicted. Table 1 gives a summary of the Standish report 2015.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

Table 1: 2015 Chaos Report by Standish Group on Software Failure

YEAR	2012	2013	2014	2015
SUCCESSFUL	27%	31%	28%	29%
CHALLENGED	49%	50%	55%	52%
FAILED	22%	19%	17%	19%

The Roles of KM in Software Engineering

Software Engineering came into existence as a discipline to solve the problem of software crisis; since its emergence, different approaches have been proposed to reduce the cost of software projects, shorten the time required for completion, and to improve quality. Process improvement, introducing new technologies and improving the people's performance (people ware) are some of the approaches been used. KM addresses mainly people – ware. This is simply because software development is a human and knowledge – intensive creative activity. KM acknowledges the importance of individuals having access to certain information and knowledge when they need to complete a task or make a decision (Ioana and Mikael, 2002).

However, KM does not ignore the other software development aspects, such as process and technology; neither does it seek to replace them. Instead, KM works toward addressing the management of organizational knowledge. The acquisition, storage, organization and effective access to this knowledge are of utmost importance in Knowledge Management.

KM can be viewed as a risk prevention and mitigation strategy, because it addresses explicitly the risk that most organizations often ignore to address. Some of such risk factors are, as stated by Ioana and Mikael, (2002) as follows:

- Loss of Knowledge due to loss of personnel (attrition).
- People repeating mistakes and performing rework because there are no cultural practices that encourages the sharing knowledge of lessons learned from previous projects or because they forget what they learnt.
- Individuals who own key knowledge are becoming unavailable.
- Lack of knowledge, as a result of difficulties in acquiring new knowledge and skills in terms of time and cost.

Knowledge can be transferred through formal training or through learning by doing. Formal training is often time consuming and expensive and if it is done externally does not cover local knowledge. Learning by doing, can be risky because people continue to make mistakes until they get it right. KM comes in here to harmonize these two concepts by providing a pointer to people and documents from which knowledge could be derived. It does not seek to replace organized training but supports it. Knowledge that has been documented can be provided as materials for internal training courses. In the same vein, it supports learning by doing by providing pointers to people with specialized knowledge, when and where it is needed.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

KM is not expected to be performed by the software development teams themselves, but by the Chief Knowledge Officer and his staff, an Experience Factor Group or a Software Process Improvement Group. These groups of people will support the developers in their daily work rather than loading them with extra responsibilities.

The core task of the Software Engineer is to develop software. However, documents (such as contracts, project plans, and requirements and design specifications) are also produced during software development. Knowledge that is emerged from solving the project's problem can be captured and documented. This documented knowledge can be reused by the team member when faced with similar challenges in subsequent projects.

In summary, document management, competence management and software reuse are knowledge management activities that support development processes.

METHODOLOGY

This section presents the architectural design, conceptual model, the interface and database design of the Experience Repository.

Architectural Design of Experience Repository

Figure 3 depicts the architecture of the proposed KM system for ER. It shows the overall structure of the system, including the components or sub systems, their relationships, and how they are to operate. The following documents the knowledge to be captured from software development team members and projects' artifacts.

System Analyst (SA): SA is a member of the development team that is responsible for the analysis of the requirement of the intended system to be developed and in most cases also see to the overall phases of the System Development Life Cycle (SDLC). He contributes and deposits his experience into the ER in order to help other programmers or developers to tap into their experiences and to learn from each other.

System Designer (SD): The system designer is responsible for the design of the architectural framework of intended project or system. He also contributes and deposits his experience into the ER in order to help other designers to learn from his project's experience.

Programmer: This is the member of the development team that is responsible for writing the codes and implementing the design. He also contributes into ER, by making his codes available for the code to be reused when similar task is to be accomplished in the nearest future.

Data, Process and Documents: These are project products that are produced by the developers in the cause of their work. These products usually include documents, UML models, SQL tables, executables, code libraries, and so on. All these are properly organized in the ER and made available for new or less experienced developers.

New Developer: A new developer represents any member of the organization that may need the resources made available in the ER. He / She is supposed to navigate the ER for relevant information, access and utilize the information found relevant and also deposit his / her own

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

experience (contributions, observation or modification as it may be required) that must be considered worthy to be in the repository by the administrator of the ER.

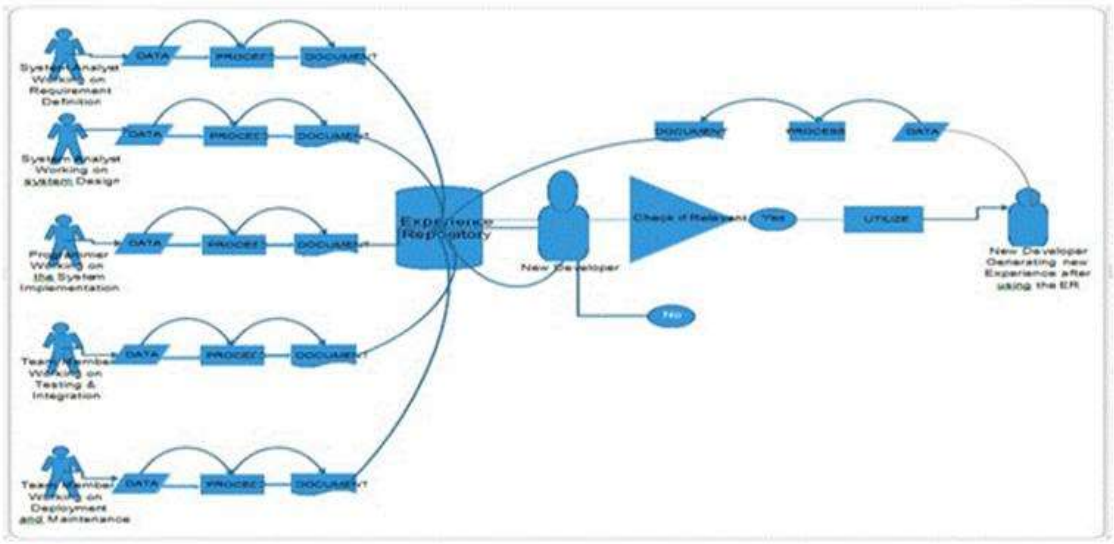

Figure 3: Architectural Design of Experience Repository

Conceptual Modeling of the Experience Repository

The Conceptual Model in figure 4 is used to structure or represent software projects' knowledge for the Experience Repository. The project managers and project developers that deposit lesson learnt in the repository as experiences and the new developers that make use of the experience are referred to as users. A user creates projects or adds contribution to project development. Other entities such as Project and contributions with respective attributes amongst others are as shown in figure 4.

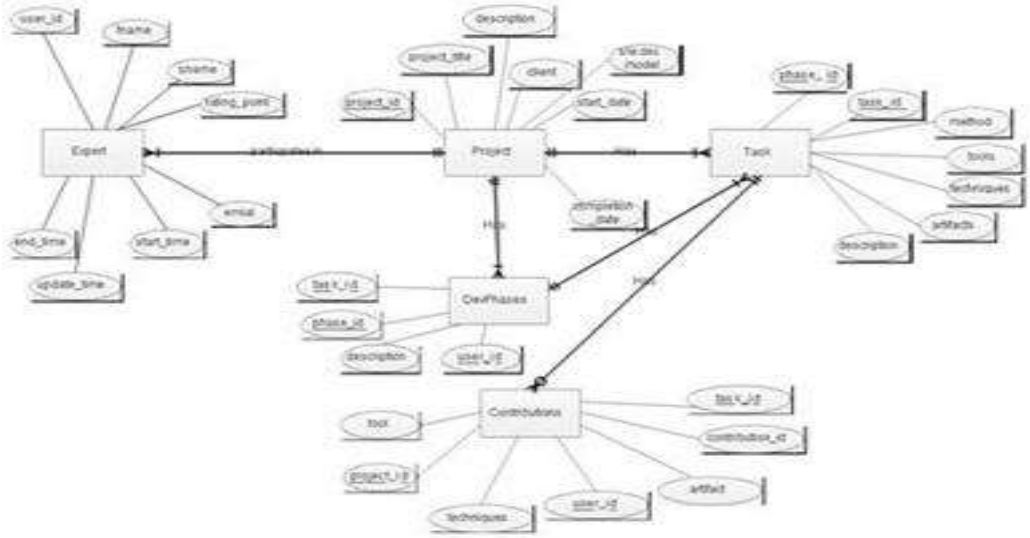

Figure 4: Conceptual Model for Experience Repository

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

Document Management Technique for Experience Repository

Software firms usually have vast amount of information that are required for their on – going projects or for their future projects in the form of knowledge of their main actors or in documents. But lack of information sharing among practitioners and various project groups, lack of good management of information assets and lack of support from the knowledge make this information not available and not useful. Hence, the need for a system that could cater for this requirement make Document Management an effective tool for knowledge management.

Document Management Process

A process has to be in place when a document goes from the experienced developer (Creator) to new developer (consumer); this is to ensure that everything works as planned and according to expectation. Figure 5 illustrates the process of converting data into knowledge.

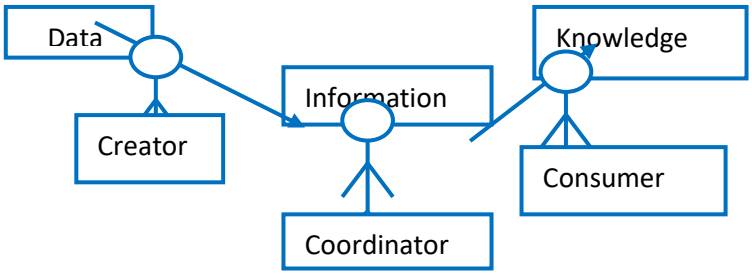

Figure 5: Process of Conversion of Data into Knowledge

Document Management Technologies (DMT)

The Experience Repository created in this work makes use of the following Document Management Technologies such as: repository, conversion, indexing and searching, creation and distribution (Meyers and Jones, 1999)

Repository

This is the place where documents/objects are kept and restored for use. Database Management System (DBMS) are the most commonly used repositories. The DBMS serve as the backend, a server in the middle and a client system as the front end.

Conversion

Most of the documents are comprised of text, images and multimedia objects. Since each of these formats are fundamentally different, they have to be converted using different tools and techniques. The main importance of conversion is to help to improve the performance of searching and retrieval of stored documents.

Indexing and Searching

The following sections introduce the search methods adopted in this work.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

Indexing

As the documents growth increases in the Experience Repository, the speed of information retrieval becomes critical. Indexing allows the breaking up of documents into more granular down to word level. One popular way of indexing is **Inversion of Terms**. This method will have a sorted list of all the keywords in a file with pointers pointing to the actual location.

Searching

Secondary search: Search within the results of the first search The ability of an Electronic Document Management System (EDMS) to provide the searching facility to the user is explore in the design of the Experience Repository used for knowledge Management in this work. The user would be able to search for a particular document or a document containing particular information without delay. Information retrieval effectiveness is measured by recall and precision. Recall is the proportion of the relative materials retrieved and precision is the proportion of retrieved material that are relevant. The following are types of search mechanisms used.

Semantics: Identify the exact definition of the word and then start searching.

Synonym: When searching for a particular word, search also for its synonyms (e.g Searching for female will search for woman or girl)

Boolean: is the ability to merge together multiple words with operators like AND or OR

Proximity: is the ability to search for multiple words grouped together.

Fuzzy word: is used when the word is not perfect due to some reason (OCR reading). This uses character and string pattern searching to determine the correct word.

Concept: is the ability to pass information or news related to a concept

Creation

Document Creation can be viewed as the creation of a container, which brings together information from a variety of sources, in a number of formats, around a specific domain/topic to meet the need of a particular individual or an organization.

Workflow

This is a way of tracking the state of the document and who is responsible for that step. A workflow comprise of four primary elements, namely, process, actions, people and document.

Process: This is the sequence of steps necessary to reach a desired objective. The process can be structured or ad-hoc.

Actions: This refers to what is to be done at each stage

People: These are the actor who is to accomplish a specified task. This rather specified by the roles and not by individuals.

Document: This is the focus of the process. In our context, it refers to the project products that are produced by the developers in the cause of their work. These products usually include

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

project artifacts such documents, UML models, SQL tables, executable codes, code libraries and so on.

Distribution

Distribution is the act of delivering the needed information, usually in document form, to the end user who will use or consume it. The format of the document will vary based on the organizational style and user needs.

All these DMT was applied to knowledge exploitation from the ER. It helps to manage large number of engineering documents and operating procedures. Software development team members will have immediate access to the current revision of every document allowing rich collaboration between them. The development of a prototype of ER includes this technique.

Implementation of Experience Repository for SPI

The proposed architecture of Experience Repository was implemented as a KM system prototype to validate the proposition that software crisis could be mitigated with the availability of relevant project experience and artifacts.

The ER developed in this work is a collaborative tool for developers and project managers to share experience. Both experienced and inexperienced developers find the tool useful to support their work. It is a web - based application.

The Rating Mechanism

The software provides a rating mechanism in form of a Dashboard. The importance of the dashboard is to encourage practitioners to make use of the ER. Usually the software developers are very busy people who always concern themselves with delivery of project within a very tight time schedule. Hence, they may not be interested in populating the ER for future use. Since software organizations are not only interested in the current project but also in capturing and documenting the knowledge that emerges from solving the project's problem for future use.

We introduce the rating mechanism to respond to some limitations of ER found in literatures. Finn and Tor (2005) claimed that an ER that is not used by software developers is of no value to the company. They stated further that: "The ER must contain a minimum amount of experiences that can be searched. The amount of experience available is critical. If there is little experience available in the ER, the developers will neither use it nor contribute their own experiences to it". Kelvin (2003) and Michael (2014) hold similar views.

Hence, this work introduces the technique of the dashboard to actually motivate the developers to make use of the ER. The points scored by each developer can be used to provide incentives to them based on the organizational policy of the firm.

After a successful registration, the system will take the user to the dashboard showing his/her name, the number of projects created, and the contributions made. Points are allotted for using the ER. By default, the dashboard will show zero, but after creating a project successfully or a contribution is shared, the number of projects is incremented by "1", and a user is given 100 points per project. This is shown in figure 5.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

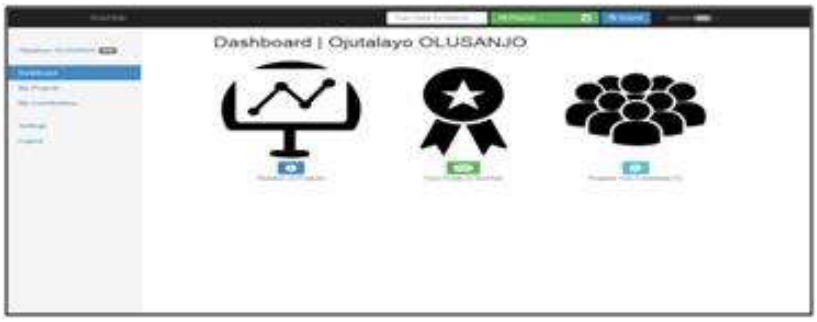

Figure 5: Imo –Hub Dashboard

Acquisition and Reuse of Software developers' Experience and Project Task

A document template is created to add the project task. At this template the developer can select the project task he intends to perform, that is, the development phase in which the task fall into. The template is accessed from the dashboard as a user clicks on current project. The template, as depicted in figure 6, has the facility to document the tools, the techniques, and other artifacts used during development of project.

Figure 6: Project Task Template of "Imo – hub"

The experience repository provided a platform for acquisition and reuse of software developers' experience and projects' processes, artifacts, documentation. The system has provided a formal and organized method of knowledge creation and transfer.

CONCLUSION

The availability of an ER will help to mitigate the problem of knowledge loss due to attrition, individual owning key knowledge becoming unavailable, as well as to facilitate knowledge reuse to keep the cost of project within budget and schedule which should be delivered with high level of customers' satisfaction.

There should be motivation for the developers to make use of the ER. The tool developed for Knowledge Management in this work incorporated a rating mechanism to show how well a particular developer participated in the Knowledge Management Initiative (KMI). The rating could be used by software firms for commendation, end of the year award or even elevation

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

of developer who attains a particular threshold of participation. This will ensure that the ER contains sufficient and up to date experiences.

Further research work entails determining what constitutes a worthwhile lesson learned and discovery of non-trivial trends in ER for facilitating strategic decision making by Software projects managers.

REFERENCES

Abran, A., Moore, W., Bourque, P., Dupuis, R., and Tripp, L. (2004). Guide to the Software Engineering Body of Knowledge. IEEE. *ISBN 0-7695-2330-7*.

Agresti, W. (2000) "Knowledge Management", *Advances in Computers*, Vol. 53, No. 7, pp 171 – 283.

Alan, F. M. (2014) "A Synthesis of Knowledge Management Failure Factors",, www.knowledge_management_tools.net.

Davenport, T. H. (1994) "Saving IT's Soul: Human Centered Information Management", *Harvard Business Review*, Vol. 72, No.2, pp 119 -131.

Davenport, T. H. and Prusak, L. (2000) *Working Knowledge: How Organizations Manage What they Know*. Harvard Business School Press, Boston, M.A.

Finn, O. B. (2007) "Knowledge Management in Software Process Improvement", Doctoral Thesis. Department of Computer and Information Science, Faculty of Information Technology, Mathematics and Electrical Engineering. Norwegian University of Science and Technology.

Finn, O.B. and Tor, S. (2005) "Harvesting Knowledge through a Method of framework in an Electronic Process Guide", Paper Read at 7th International Workshop on Learning Software Organization (LSO).

Ioana, R. and Mikael, L. (2002) "Knowledge Management in software Engineering", www.grovo.com/knowledge_management downloaded on 21st May, 2014.

Mathiassen, L., Pries-Heje, J. and Ngewenyama, O. (2001) *Improving software organizations – From principle to Practice*, Addison press, Wesley.

Michael, E.D.K. (2012) "Knowledge Management Explained" *Knowledge Management World Magazine*, [online], www.kmworld.com

Mikael, L., Ioana, R. J., Rikin, T. (2001) "A state of the Art Report: Software Tools for Knowledge Management", Fraunhofer Center for Experimental Software Engineering Maryland and University of Maryland, College Park, Maryland.

Meyes Scott and Jones Jason, "Document design for effective electronic publication", Proceedings of the 5th Conference on Human Factors and the Web, June 1999.

Nonaka, I. (1991) "The Knowledge Creating Company", *Harvard Business Review*, Vol. 6, No. 69, pp 96–104.

Nonaka, I. and Takeuchi, H. (1995) *The knowledge creating company: how Japanese companies create the dynamics of innovation*, Oxford University Press, Cambridge.

Oladejo, B. F., Odumuyiwa, V. T. and David, A. A. (2010) "Dynamic Capitalization and Visualization Strategy in Collaborative Knowledge Management System for EI Process" Paper read at International Conference in Knowledge Management and Knowledge Economy, Paris, June.

Pressman, R. S. (2010) *Software Engineering: A Practitioner Approach*, 7th Ed. Thomas Caisson, New York.

Paulish, D. (1993) *Case Studies of Software Process Improvement Methods*, Software

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)

Engineering Institute. Carnegie Mellon, USA.
Rowe, A. (2006) "Looking through the Chinese 'lens' of corporate environmental management", *Journal of International Business Strategy*, Vol. 4, No. 1, pp 105–112.
Salina, D. and Wan-Fadzilah, (2008). " An Empirical Study of Knowledge Management Process in Medium Enterprises", *Communication of the IBIMA*, Vol. 4, pp 169-177
Sommerville, I. (2007) *Software Engineering*, 8th Edition, Pearson, New York.
Sommerville, I. (2009) *Software Engineering*, 9th Edition, Pearson, New York.
Sven, A C. and Mikael, S. (2003) "Software Process Improvement through Knowledge Management", www.grovo.com/knowledge_management
Wenger, I. and Etienne, C. (1998) *Communities of practice: Learning, meaning and Identity*, Cambridge University Press. UK.
Wenger, I., Etienne, C. and Snyder, W. M. (1999) "Communities of practice: The Organizational Frontier", *Harvard Business Review*, Vol. 78, No. 1, pp 139-145.
Yin, R.K. (2003) *Case Study Research: Designs and Methods*, 3rd Edition, Sage Publications, Thousand Oaks, CA.

ISSN 2054-0957 (Print), ISSN 2054-0965 (Online)