© 2019 *Afr. J. MIS*.
https://afrjmis.net

A Generic Framework for Software Process Knowledge Harvest of Human Level Activities

R. O. Oveh[1] and I. C. Ifediora[2]
[1,2]Department of Mathematics and Computer Science,
Western Delta University, Oghara, Delta State, Nigeria.
Email: [1]omo_rich@yahoo.com
[2]Ifylaw31@gmail.com
&
F. A. Egbokhare
Department of Computer Science,
University of Benin, Benin City, Nigeria.
Email: fegbokhare@uniben.edu

ABSTRACT

The threat of tacit knowledge loss in software development process necessitates the need for its harvest. Knowledge harvesting is the process of capturing domain knowledge process, culture and practices. It builds explicit and formal knowledge capacity. The reuse of harvested knowledge gives organisations comparative advantage. This paper harvested software process knowledge of human level activities from 4 software development organisation. The harvested knowledge was used in creating a generic framework for software process. A questionnaire was used to validate the knowledge obtained. The validation metrics used in the questionnaire are: vocabulary, clarity, conciseness, coverage and completeness. The result showed that the knowledge obtained is consistent with the knowledge in the domain and it is suitable for reuse.

Keywords: *Human level activities, Software process, knowledge, Tacit Knowledge, Knowledge harvesting Asymmetrical*

Reference Format:

Oveh, R. O., Ifediora, I. C. and Egbokhare, F. A. (2019), A Generic Framework for Software Process Knowledge Harvest of Human Level Activities,
Afr. J. MIS, Vol.1, Issue 4, pp. 69 - 86.

1. INTRODUCTION

Domain knowledge is the driver for any domain. Its availability and reuse ensures productivity in any domain (Oveh et al., 2019). Domain knowledge is knowledge about a domain that is acquired by formal or informal training (Josh, 2017). Domain knowledge could be explicit or tacit. Explicit domain knowledge is the knowledge about a domain that is formally documented. Such knowledge can be easily transferred from one domain user to another because it is formally documented. The bulk knowledge of a company is in its tacit knowledge. Tacit domain knowledge is implicit and inherent in a person in the domain (Hajric, 2018). Tacit knowledge is usually resident in a domain expert, who has acquired such knowledge with experience on the job. The knowledge is on the process, culture or practices of the domain, which is domain specific. There is the threat of this knowledge being lost in a case of the person departing from an organization as a result of: sack, retirement, sickness, natural disaster or even death (Lukang, 2018). This kind of knowledge can be obtained from the individual by harvesting.

According to Serrat (2017), If 80% of knowledge is unwritten and largely unspoken, we first need to elicit that before we can articulate, share, and make wider use of it. He defined knowledge harvesting as a means to draw out, express, and package tacit knowledge to help others adapt, personalize, apply it, build organizational capacity, and preserve institutional memory. Knowledge harvesting can be applied to any field of human activity. In organizations as well as society, opportunities lie in operations, products, services, strategies, and even management. Knowledge harvesting deliberately elicits, organizes, packages, and shares know-how. Several intra-organizational factors drive knowledge-harvesting design: the principal being (i) tacit knowledge enablers and inhibitors; (ii) the criticality of the knowledge to the organization; (iii) the need for immediate transfer; (iv) the complexity of the knowledge topic; (v) the qualities of knowledge contributors; (vi) the characteristics of knowledge seekers; (vii) the dispersion of knowledge contributors and knowledge seekers; (viii) the type of facilitation required; and (ix) the need for external review and validation.

Knowledge harvesting is not a catchall solution but an integral part of Knowledge Management (KM). Knowledge management is essentially about getting the right knowledge to the right person at the right time. This in itself may not seem so complex, but it implies a strong tie to corporate strategy, understanding of where and in what forms knowledge exists, creating processes that span organizational functions, and ensuring that initiatives are accepted and supported by organizational members (Mansoori et al., 2018). KM as defined by Brito et al., (2019) is the set of practices and initiatives for managers to create, transfer, and apply knowledge in organizations. Knowledge archiving and management are key concepts of ensuring long-term digital storage of conceptual blueprints and specifications of systems, methods and frameworks with capacity for future updates while making such information readily accessible to relevant stakeholders on demand (Fagbola et al., 2019). It may solely focus on knowledge sharing, storage, and refinement (Hajric, 2018). Proper capitalization of knowledge determines competitiveness and long-term sustainability. As a clear sign of this trend, knowledge management has emerged as a key interest in both academia and practice, being immersed into various fields and disciplines (Zbuchea et al., 2019).

The software development process consists of a set of human – level activities where knowledge is created and utilised. Turaga et al., (2008) described activity as complex that involves coordinated actions among a small number of humans. Aggarwal and Ryoo (2011) categorized human activities into various types, depending on their complexity. They conceptually categorized human activities into four different levels: gestures, actions, interactions, and group activities. Gestures are elementary movements of a person's body part, and are the atomic components describing the meaningful motion of a person. "Stretching an arm" and "raising a leg" are good examples of gestures. Actions are single-person activities that may be composed of multiple gestures organized temporally, such as "walking," "waving," and "punching."

Interactions are human activities that involve two or more persons and/or objects. For example, "two persons fighting" is an interaction between two humans and "a person stealing a suitcase from another" is a human-object interaction involving two humans and one object. Finally, group activities are the activities performed by conceptual groups composed of multiple persons and/or objects: "A group of persons marching," "a group having a meeting," and "two groups fighting" are typical examples. This paper is focused on harvesting actions, interactions and group activities for software process knowledge, with the aim of software process improvement.

2. LITERATURE REVIEW

Software processes are all the processes that are used to create software, including analysis, programming, configuration management, etc. (Kneuper, 2001) Software process is a knowledge driven and knowledge intensive process that involves several other sub-processes. Software process can also be defined as the set of related activities that are used in developing software. Knowledge in Software Engineering (SE) is diverse and organizations have problems capturing, retrieving, and reusing it. An improved use of this knowledge is the basic motivation and driver for Knowledge Management (KM) in SE (Rus and Lindvall, 2002). Harvesting, representing and reusing knowledge within a domain leads to maximum payoff, which is desirable in most organisations (Oveh and Egbokhare, 2019). Knowledge Management (KM) is defined as an effort to capture critical knowledge and share it within an organization (Davenport and Prusak, 1998; Alavi and Leidner 2001). It capitalizes on the collective organizational memory to improve decision making, enhance productivity, and promote innovation (Taluja et al., 2010; Perez, 1999). It is also the process of transforming information and intellectual assets into persisting value. KM connects people with the knowledge that they need to take action, when they need it (Kidwell et al., 2000).

Knowledge management involves the identification and analysis of available and required knowledge (Firestone, 2001). It helps an organization to gain insight and understanding from its own experience. Specific knowledge management activities focus on acquiring, storing and utilizing knowledge for problem solving, dynamic leaning, strategic planning and decision making. This prevents intellectual assets from decay, adds to a firm's intelligence and provides increased flexibility (Abdul-Kalam, 2004).

SE comprises several interrelated subdomains such as Requirements, Design, Coding, Testing, Project Management, and Configuration Management. There are several software process models which describe the sequence of activities carried out in developing software. These software process models are a standard way of planning and organizing a software process. The major phases are requirement gathering, design and coding, implementation and maintenance.

The management of knowledge and experience are key means by which systematic software development and process improvement occur. Within the domain of Software Engineering (SE), quality continues to remain an issue of concern. Knowledge Management (KM) gives organizations the opportunity to appreciate the challenges and complexities inherent in software development (Ward and Aurum, 2004).

Successful organisations continuously improve their processes. Like organisational standard process definition, systematic process improvement is more effective and efficient if it is done guided by process quality models and standards. The purpose of most standards is to help software organisations achieve excellence by following the processes and activities adopted by the most successful organisations (Falbo and Bertollo, 2009).

Knowledge harvesting can be applied in different areas. For example, Oveh et al., (2019) harvested result processing domain knowledge from a Higher Education Institution, using University of Benin as a case study. Semi structured interview and observation were the instruments used. Interviews were conducted to understand and represent the key elements of result processing (i.e. the who, what and how). The interviews were geared towards identifying the people involved in result processing and how the various components 'who', 'what' and 'how' interact. Implicit and explicit knowledge in result processing were harvested and represented informally using a concept map which was then modelled. Ralf (2001) did an overview of the knowledge needed by developers. He reviewed various approaches to knowledge representation including artificial intelligence, structured plain text, and software process modeling approaches, to identify representation schemes suitable for software processes. He also looked at the management aspects of software process knowledge.

Meeham and Richrdson (2002) observed that while knowledge exists in the software development process, this knowledge is not made explicit within the organization. This view is upheld by O'Conor and Lero (2014) though in their opinion, knowledge from traditional software development process activities is more explicit than that from Agile methods which is considered tacit. Managing these knowledge assets has therefore become critical for software development organizations to gain competitive advantage especially in today's knowledge intensive business environment. Bjørnson and Dingsøyr, (2008) reviewed empirical studies of knowledge management initiatives in software engineering, The majority of empirical studies relate to technocratic and behavioural aspects of knowledge management, while there are few studies relating to economic, spatial and cartographic approaches.

A finding reported across multiple papers was the need not to focus exclusively on explicit knowledge, but also to consider tacit knowledge. Mitchell (2012) advanced the use of KM in Software Process Improvement by investigating knowledge flow within an individual software development project as it executes and impediments to that flow. One of the key findings from the study was the need for management of both explicit and tacit project-level knowledge.

Oveh and Egbokhare (2018) studied the level of knowledge management in Nigerian Software Development organizations. The Findings from the study showed that though there is no formal knowledge management culture during the software process, it was observed that there are certain human level activities that can be harvested to create knowledge assets. Balanzo-Guzman (2017) harvested knowledge stances. He discussed institutional work and innovation intermediation streams of literature. He also reflected deeper on their tenets, pointing at overlapping and complementary features to be found on its foundations. He showed a strong focus on mesolevel agency, sketching on its relations to practice, boundaries and institutions. He further proposed an analytical synthesis, in an attempt to define a set of concepts likely capturing agency. The synthesis discusses possible repertoires played by actors, labelled as knowledge stances.

3. METHODOLOGY

A case study methodology was used to harvest knowledge for software development process in this study. The aim was to seek knowledge produced from human level best practices in a software process. A case study research method is an in depth study of a particular situation that provides more realistic responses than a purely statistical survey (Shuttleworth, 2017). Yin (1984) defines the case study research method as an empirical inquiry that investigates a contemporary phenomenon within its real-life context; when the boundaries between phenomenon and context are not clearly evident. Mesec (1998) in Starman (2013) suggests that a criterion for choosing an object for a case study is to consider where a practical problem in the domain the researcher is interested in exists. Since the focus of this research is to harvest knowledge about best practices for the software development process, the following guidelines aided in the selection of the organizations for the study:

1. The organization must be into software development
2. The organization must have consistently handled software development projects over the last 10 years
3. There should be existing records of successes and failures in software development projects
4. Willingness to share information about best practices and how they are retained within the organization
5. Access to staff for the purpose of data collection

CASE STUDY 1
The first case is an indigenous software development organization with head office in Abuja and branch offices in Lagos and Makurdi. The company was established in 2004 and since then they have developed various categories of software systems such as

i. Human Resource systems
ii. Enterprise solutions
iii. Payroll systems for private and public organizations
iv. Portal for Managing Academic and Administrative processes in Tertiary Institutions
v. Portals for secondary schools
vi. Cash flow management systems and
vii. Budgeting systems

CASE STUDY 2
Case 2 is a Multinational Software development Organization that specialises in developing:

i. Banking solutions
ii. Human Resource management systems
iii. Process Control Systems
iv. Payroll Systems
v. Security systems.
vi. Materials management systems

Their Head office is in Abuja with a branch office in Lagos. They have a vision to be the software institution of reference in Africa. With over two decades of providing robust solutions to individuals and corporate organisations, they have become a house hold name in Africa.

CASE STUDY 3
Invests in Africa's most talented software engineers to help companies solve the technical shortage and build high-performing distributed engineering teams. They bridge the gap of solving the global tech talent shortage while catalysing the growth of tech ecosystems on the African continent. Their main focus is:

1. Development of security solutions
2. Training/development of specialised teams (e.g. Java Team, .Net Team, Oracle Team)
3. Remote Software Development
4. Web Development

CASE STUDY 4

This Software Development Company provides consultancy services and online recruitment services. They develop:

i. Human resource Management Systems
ii. Third Party Online Payment Solutions
iii. Payroll and other financial systems

To validate the harvested knowledge, a questionnaire was designed with the following questions:

1. Ethnographic study in requirements gathering involves the study of?
2. What are the processes for gathering requirements from stakeholders & end-users?
3. What are the processes of software development?
4. What are the physiological processes for blocks resolution during coding?
5. What is the appropriate approach to software design?
6. What are the people and processes involved in user tasks?
7. What are the tools used in ethnographic study of stakeholders and end-users?
8. What are the stages involved in implementation?
9. What does a deployed system need for maintenance?
10. Who can business rules be obtained from in a domain?
11. How can low bus factor issues be handled in coding?
12. How can code ownership issues be resolved in coding?
13. How can knowledge be shared in software process?
14. How can knowledge be transferred in software process?

The validation metrics used in the questionnaire were: accuracy, clarity, conciseness, consistency, coverage, and completeness. These items assessed the aspects of vocabulary (i.e. the names), representation (i.e. the relationship between the structure and the semantics), semantic richness, and context coverage (i.e. the features). The context in which each metric was used are defined below:

1. Accuracy – evaluates if the ontology agrees with the domain experts knowledge.
2. Clarity - evaluates how well the ontology expresses the meaning of the terms
3. Conciseness - evaluates the ontology for redundant or irrelevant element
4. Consistency - evaluates the ontology for contradictions.
5. Coverage - evaluates the representation of the ontology with respect to the domain (i.e. how well it represents that domain)
6. Completeness - evaluates if the domain is fully covered.

Seven (7) copies of the questionnaire were given to Domain Experts for content validity. The Domain Experts were Project Managers and Senior Programmers with 10-30 years' experience. Necessary corrections were made based on the suggestions from the domain experts before the final copies were produced. Fifty (50) hard copies of the questionnaires and a Google form soft copy were administered across three geographical areas in Nigeria (i.e. South (Edo, Rivers, and Delta), North (Abuja, Kano), and West (Lagos) with the aid of research assistants. Thirty one (31) hard copies and 24 (twenty four) softcopy questionnaires were returned. Two (2) questionnaires were not properly filled and were unusable. A total of fifty three (53) questionnaire returned were used for the final analysis. The respondents were mainly software Project managers and programmers with 5-30 years' software development experience.

4. RESULT

The harvested knowledge was obtained from four core activities in a software process: Requirements definition, Design, Coding and Implementation and maintenance. The harvested knowledge was organized into the hierarchy in Figure 1. The axioms are presented in Table 1, object properties Table 2 and questionnaire response in Table 3.

Axioms

Axioms provide a proper way to add logical expressions. Such logical expressions can be used to refine the concept and their relationships. Axioms are used to design an explicit way of expression that is always true. Axioms can be used for defining the meaning of several components of a concept, defining complex relationships and verifying the correctness of the information or obtaining new information (Zeshan and Mohamad, 2012). Table 1 presents the axioms for the software process framework.

DISCUSSSION

Object properties in Table 2 specify the relationship between concepts in the framework in Figure 1. It aids in interpreting the relationship between the concepts. Reading from the Figure 1, the concept ethnographic study identifies stakeholders and user tasks. User tasks is used to determine: “who”, “what”, “how” of a domain. Maintenance requires that a system is first deployed first. Domain experts should be interviewed or observed to obtain domain knowledge and business rules. Implementation should be done through incremental process and the changeover should be phased. Design should be modular and done with mathematical process. Modular design should be implemented using data flow diagram. Block resolution should be done by going to senior/experienced programmers, problem domain, physical objects like games, interactive blogs like stackexchange or stackoverflow and definition and design.

Further maintenance should be done after a system is deployed with response from users’ comments and requirements. Requirements should be elicited from stakeholders and end-users through: observation, interview, document review, brainstorming or questionnaire. Design and coding should encourage code review, knowledge retention, unique coding and pairwise coding. It should also discourage code ownership through version control. Bus factor in a software process can be increased by clean codes, code review, pairwise coding and code documentation. The response from Table 3 shows that the domain experts strongly agree with the harvested knowledge and the framework and thus validating the framework.

5. CONCLUSION

Knowledge is an invaluable intellectual capital that can give organisations competitive advantage if harvested for reuse. The threat of tacit knowledge loss in software development process necessitates the need for its harvest. Knowledge harvesting is the process of capturing domain knowledge process, culture and practices. This paper harvested software process knowledge of human level activities from 4 software development organisation. The harvested knowledge was used in creating a generic framework for software process. A questionnaire was used to validate the knowledge obtained. The validation metrics used in the questionnaire are: vocabulary, clarity, conciseness, coverage and completeness. The result showed that the knowledge obtained is consistent with the knowledge in the domain and it is suitable for reuse. We recommend that the harvested knowledge should be formally represented using ontology.

REFERENCES

[1] Abdul-Kalam, A.P.J. (2004) Digital Library and its multidimensions. President of India’s speech at the “Inauguration of International Conference on Digital Libraries (ICDL) retrieved 16/9/18 from: http://www.presidentofindia.nic.in/scripts/sllatest1.jsp?id=282

[2] Aggarwal, J.K. and Ryoo, M.S. (2011) Human Activity Analysis: A Review. A review. ACM Computing Surveys 43(3) 1-43.

[3] Alavi, M. and Leidner, D. E. (2001). Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS Quarterly, 25(1), 107-136.

[4] Balanzo-Guzman, A. (2017) Theory within a Policy Dissecting Capacity Development, Harvesting Knowledge Stances. Italian Journal of Science & Technology Studies. 8 (1) 73-102.

[5] Bjornson, F.O. and Dingsoyr, T. (2008) Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. Elsevier Journal of Information and software technology. 50 (11) 1055-1068.

[6] Brito, L. M. P,.Alvws da Silva, N. E., Caraxo de Castro, A. B., Nodari, C, H,. Pereira da Silva, A. W. (2019) Knowledge management for the sustainable development of the semi-arid region in Northeastern Brazil. Ciência Rural, 49(4) 1-7.

[7] Davenport, T.H. and Prusak, L. (1998) Working Knowledge – How Organizations Manage What They Know. Harvard Business School Press, Boston, Massachusetts.

[8] Fagbola, T. M., Thakur, S. C., and Olugbara, O. (2019) Rab-KAMS: A Reproducible Knowledge Management System with Visualization for Preserving Rabbit Farming and Production Knowledge. International Journal of Advanced Computer Science and Applications, 10(1) 263-273.

[9] Falbo, R. A. and Bertollo, G. (2009) A software process ontology as a common vocabulary about software processes. International Journal of Business Process Integration and Management. 4(4) 239-250.

[10] Firestone, (2001) Key Issues in Knowledge Management, Knowledge and Innovation. Journal of the KMCI; 1(3), 8-38.

[11] Hajric, E. (2018) Knowledge Management System and Practices - A Theoretical and Practical Guide for Knowledge Management in Your Organization. Organizational Learning, Jacksonville, FL.

[12] Hajric, E. (2018) Knowledge Management System and Practices - A Theoretical and Practical Guide for Knowledge Management in Your Organization. Organizational Learning, Jacksonville, FL.

[13] Josh, S. (2017) How important is domain knowledge for Business Analysis. ZaranTech. retrieved 8/9/2019 from https://www.zarantech.com/blog/how-important-is-domain-knowledge-for-business-analysis/

[14] Kidwell, K.M., Vander L. and Johnson S.L. (2000). Applying corporate knowledge management practices in higher education, Journal of Educause Quarterly 4, 28-33.

[15] Kneuper, Ralf. (2001). Supporting Software Processes Using Knowledge Management. Handbook of Software Engineering and Knowledge Engineering. Vol 2 pp 579-606.

[16] Lukang, W. (2018) Domain Knowledge. Lead Change. retrieved 7/8/2019 from https://leadchangegroup.com/domain-knowledge/

[17] Mansoor,i A.A., Yazid A., Khatibi A., Azam S.M. (2018) A Scheme Of Knowledge Management Strategies and Organizational Performance In Abu Dhabi Government Entities. European Journal of Management and Marketing Studies. 1(4) 111-130.

[18] Meehan, B. and Richardson I. (2002) Identification of Software Process Knowledge Management. Software Process Improvement. 7, 47-55.

[19] Noy, N. F., and McGuiness, D. L., (2001). Ontology Development 101: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.

[20] Oveh, R. and Egbokhare, F. (2018) Preliminary studies on assessment of the level of knowledge management in Nigerian software development organisations. Nigerian journal of education, health and technology research (NJEHETR) 10, 93-99.

[21] Oveh, R.O. and Egbokhare, F.A. (2019) Harvesting and Informal Representation of Software Process Domain Knowledge. Intelligent Computing Conference, 2, 936–947. Springer Nature Switzerland.

[22] Oveh, R.O, Ifediora, I.C., & Egbokhare, F.A (2019) Domain Knowledge Harvest of Result Processing Activity in Higher Institutions of Learning. African Journal of Management Information System. 1(4) 1-14.

[23] Perez, E. (1999) Knowledge Management in the Library—Not. Database Magazine 22(2), 75–78.

[24] Rus, I., and Lindvall M. (2002) Knowledge Management in Software Engineering, IEEE Software, pp. 26-38.

[25] Sawsaa, A. F. (2015) Ontological Engineering Approach of Developing Ontology of Information Science. Anchor Academic Publishing. https://books.google.com.ng/books?isbn=3954894483

[26] Serrat, O. (2017), 'Harvesting Knowledge', in Knowledge Solutions, pp. 1069 – 1075, Springer; https://link.springer.com/chapter/10.1007/978-981-10-0983-9_122.

[27] Shuttleworth, M. (2017) The case study research design: have evolved over the past few years as a useful tool for investigating trends and specific situations in many scientific disciplines. Retrieved 25/2/17 from https://explorable.com/case-study-research-design

[28] Starman, A. B. (2013). The Case study as a Type of qualitative Research. Journal of Contemporary Educational Studies. 64 (1) 28 – 43.

[29] Taluja, R.K., Tewari, C.K. and Kaur, A. (2010). Concept of Knowledge Management and Its Usage in Higher Learning Institutions. VSRD-TNTJ. I (4), 255-265.

[30] Turaga, P., Chellappa, R., Subrahmanian, V.S. and Udrea (2008) Machine Recognition of Human Activities: A Survey. IEEE Transactions on Circuits and Systems for Video Technology. 18 (11) 1473-1488.

[31] Ward, J. and Aurum, A. (2004) Knowledge Management in Software Engineering - Describing the Process. Proceedings of the Australian Software Engineering Conference (ASWEC'04).

[32] Yin, R. K. (1984). Case study research: Design and methods. Newbury Park, CA: Sage.

[33] Zbuchea, A., Florina Pînzaru, F., Busu, M., Stan, S and Bârgăoanu, A. (2019)Sustainable Knowledge Management and Its Impact on the Performances of Biotechnology Organizations. https://www.mdpi.com/journal/sustainability., Basel, Switzerland. 11(539) 1-20.

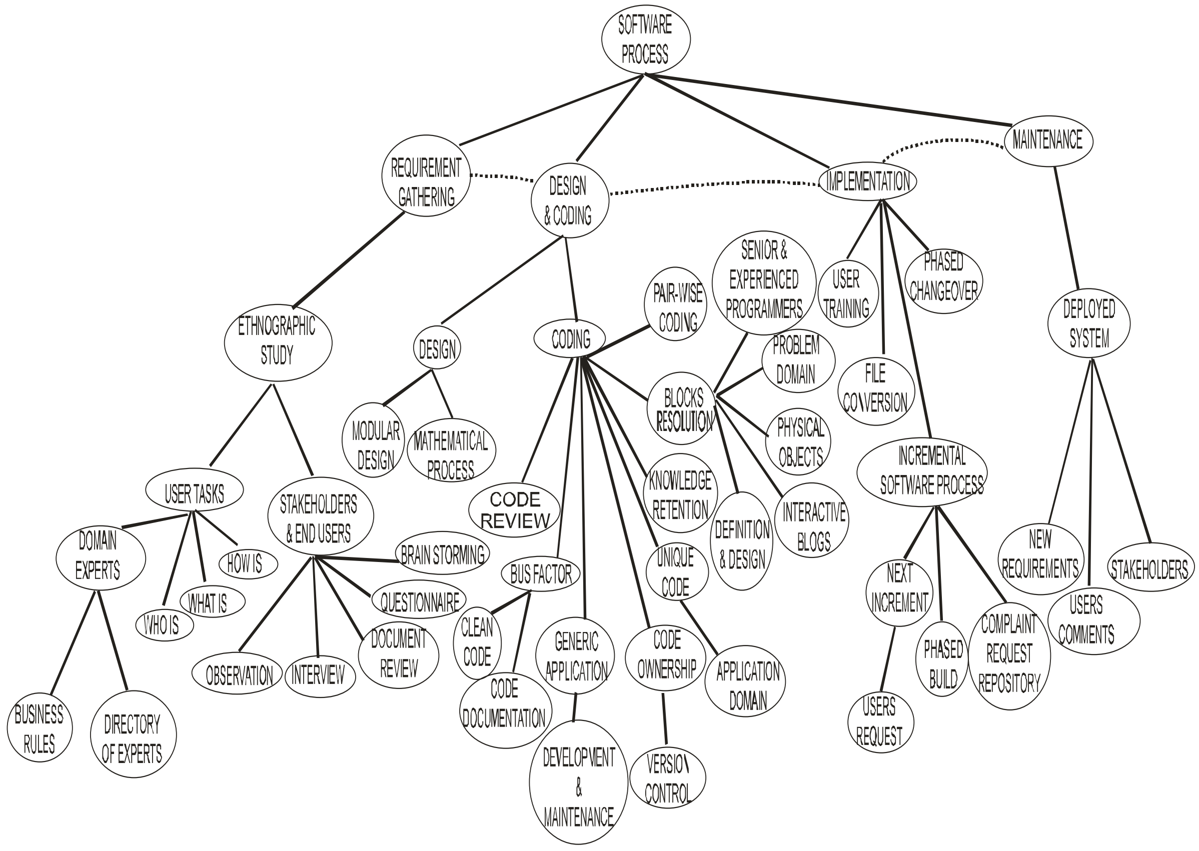

Figure 1: Framework for Software Process Knowledge

Table 1: Logical Axioms Table

Concept name	Axiom description	Logical expression
SOFTWARE PROCESS	A process or related activities that lead to the activity of producing software. The activities include: requirement gathering, design/coding, Implementation and maintenance.	$\forall X$, SOFTWARE PROCESS(X) $\Rightarrow$ ($\exists$ **requirement gathering**) and ($\exists$ **design/coding**) and ($\exists$ ***implementation***) and ($\exists$ **maintenance**)
REQUIREMENT DEFINITION		
DESIGN & CODING		
IMPLEMENTATION	Requirements are contained in the application domain using informal interviews, participant observation, questionnaire, documents review and other suitable requirements elicitation techniques.	$\forall X$, REQUIREMENT DEF(X) $\Rightarrow$ ($\exists$ **interviews**) and ($\exists$ **participant observation**) and ($\exists$ **questionnaire**) and ($\exists$ **documents review**)
MAINTENANCE		
BLOCKS		
INCREMENTAL SOFTWARE PROCESS		$\forall X$, DESIGN & CODING (X) $\Rightarrow$ ($\exists$ **pair wise coding**) and ($\exists$ **blocks resolving**) and ($\exists$ **code repository**) and ($\exists$ **theme coding**) and ($\exists$ **code review**) and
CODE REVIEW	The needed software is designed and coded using the requirements obtained.	
CODE OWNERSHIP		
VERSION CONTROL		
CODE COHESION		
MAINTENABLE CODE	The software is put into	

TIGHT COUPLING UNIT TESTING TECHNICAL DEBT COMPLAINT REQUEST REPOSITORY HOW IS WHO IS WHAT IS PAIR WISE CODING GENERIC APPLICATIONS/ THEME CODING ETHNOGRAPHIC STUDY KNOWLEDGE RETENTION INTERACTIVE BLOGS BUSINESS RULES STAKEHOLDERS CODING PROBLEM DOMAIN PROGRAMMER SENIOR PROGRAMMER	operation in the target domain Includes error correction, adapting it to a new environment or system enhancement by adding new features. Confusion and certain negative feelings during coding that may disrupt the smooth flow of the coding process The system is developed in versions or increments. The activities of requirement definition, design, coding and implementation are interwoven. A system of checking source code that is usually done by pair programmers A situation where the code is owned by the team and not an individual. It can be	($\exists$ **retreat coding**) and ($\exists$ **mentoring**) and ($\exists$ **collaboration**) and ($\exists$ **communication**) $\forall x$, IMPLEMENTATION(X) $\Rightarrow$ ($\exists$ **incremental software**) and ($\exists$ phased change over) and ($\exists$ **user training**) $\forall x$, MAINTENANCE (X) $\Rightarrow$ ($\exists$ **error correction**) and ($\exists$ new requirements) and ($\exists$ **uniform resource update**) $\forall x$, BLOCKS (X) $\Rightarrow$ ($\exists$ **definition and design**) and ($\exists$ problem domain) and ($\exists$ **senior/experienced programmers**) and ($\exists$ interactive blogs) and ($\exists$ physical objects) $\forall x$, INCREMENTAL SOFTWARE PROCESS (X) $\Rightarrow$ ($\exists$ **requirement gathering**) and ($\exists$ **design/coding**) and ($\exists$ ***implementation***) $\forall x$, CODE REVIEW (X) $\Rightarrow$ ($\exists$ **code**) and

BUS FACTOR	changed or edited by any body	($\exists$ **pair programmers**)
CLEAN CODES CODE DOCUMENTATION	A system that manages changes to document or source code. It is a repository of codes	$\forall \boldsymbol{X}$, CODE OWNERSHIP (X) $\Rightarrow$ ($\exists$ **code**) and ($\exists$ **coding team**)
	Measures the strong relationship between code that aids reuse.	$\forall \boldsymbol{X}$, VERSION CONTROL (X) $\Rightarrow$ ($\exists$ **code**) and ($\exists$ **repository**) and ($\exists$ **maintenance**)
	A code that can be easily modified	$\forall \boldsymbol{X}$, CODE COHESION (X) $\Rightarrow$ ($\exists$ **code**) and ($\exists$ **relationship**) and ($\exists$ **reuse**)
	A situation where components within a software depend on one another	$\forall \boldsymbol{X}$, MAINTENABLE CODE (X) $\Rightarrow$ ($\exists$ **code**) and ($\exists$ ***maintenance***)
	A method of testing individual source code or the smallest part of an application	$\forall \boldsymbol{X}$, TIGHT COUPLING (X) $\Rightarrow$ ($\exists$ **software components**) and ($\exists$ ***relationship***)
	It is a concept that shows the extra effort needed for a code that was designed using an easy solution instead of a better approach	$\forall \boldsymbol{X}$, UNIT TESTING (X) $\Rightarrow$ ($\exists$ **code**) and ($\exists$ ***testing***)
		$\forall \boldsymbol{X}$, TECHNICAL DEBT (X) $\Rightarrow$ ($\exists$ **code**) and ($\exists$ ***design approach***)
	A database for	

	users complain Who is responsible for certain task	$\forall x$, COMPLAINT REQUEST REPOSITORY (X) $\Rightarrow (\exists$ **users complain**$)$
	How tasks are performed	$\forall x$, HOW IS (X) $\Rightarrow (\exists$ **person**$)$ and $(\exists$ ***tasks***$)$
	The existing rules in the domain	
	A coding style where senior programmers work with younger programmers	$\forall x$, WHO IS (X) $\Rightarrow (\exists$ **task process**$)$ $\forall x$, WHAT IS (X) $\Rightarrow (\exists$ **domain rules**$)$
	Applications that use the same code with different features	$\forall x$, PAIR WISE CODING (X) $\Rightarrow (\exists$ **coding**$)$ and $(\exists$ ***senior programmers***$)$ and $(\exists$ ***junior programmers***$)$
	Participant observation of the application domain	$\forall x$, GENERIC APPLICATIONS/ THEME CODING (X) $\Rightarrow (\exists$ **Application**$)$ and $(\exists$ ***code***$)$
	Structure for keeping knowledge in an organization	$\forall x$, ETHNOGRAPHIC STUDY (X) ***participant***) and $(\exists$ ***observation***$)$ and $(\exists$ ***application domain***$)$
	A blog where questions can be asked and answered like stackoverflow and stackexchange	$\forall x$, KNOWLEDGE RETENTION (X) $\Rightarrow (\exists$ ***knowledge***) and
	Rules that govern the	$(\exists$ ***mentoring***$)$ and

	operation of a business	$(\exists\ \textit{pair wise coding})$
	A person that initiates the development of a software (management) or a person that uses the software	$\forall X$, INTERACTIVE BLOGS (X) $\Rightarrow (\exists\ \textit{blog})$ and $(\exists\ \textit{question})$ and $(\exists\ \textit{answers})$
		$\forall X$, BUSINESS RULES (X) $\Rightarrow (\exists\ \textit{rules})$ and $(\exists\ \textit{operation})$ and $(\exists\ \textit{business})$
	The process of creating an application	
	The area that needs to be examined	$\forall X$, STAKEHOLDERS (X) $\Rightarrow (\exists\ \textit{management})$ and $(\exists\ \textit{end user})$ and $(\exists\ \textit{software})$
	A person who writes codes to create software	$\forall X$, CODING (X) $\Rightarrow (\exists\ \textit{code})$ and $(\exists\ \textit{Application})$
	A person who is experienced in software development	$\forall X$, PROBLEM DOMAIN (X) $\Rightarrow (\exists\ \textit{who is})$ and $(\exists\ \textit{what is})$ and $(\exists\ \textit{how is})$
	The number of team members whose absence or exit can affect a software project	$\forall X$, PROGRAMMER (X) $\Rightarrow (\exists\ \textit{code})$ and $(\exists\ \textit{software})$
	A simple code that can be easily understood or edited	$\forall X$, SENIOR PROGRAMMER (X) $\Rightarrow (\exists\ \textit{code})$ and $(\exists\ \textit{programmer})$ and $(\exists\ \textit{software})$
	The process of writing the	

	reason for a code to make it readable for easy maintenance	$\forall \boldsymbol{x}$, BUS FACTOR (X) $\Rightarrow (\exists\ \boldsymbol{team\ members})$ and $(\exists\ \boldsymbol{programmer})$ and $(\exists\ \boldsymbol{software\ project})$ $\forall \boldsymbol{x}$, CLEAN CODES (X) $\Rightarrow (\exists\ \boldsymbol{code})$ $\forall \boldsymbol{x}$, CODE DOCUMENTATION (X) $\Rightarrow (\exists\ \boldsymbol{code})$ and $(\exists\ \boldsymbol{maintenance})$

Table.2: Object Property Definition

s/n	Object Property	Domain	Range
1	Identifies	Ethnographic study	• Stakeholders, • Users tasks
2	Todetermine	Users tasks	• Who is • What is • How is
3	Require	Maintenance	• Deployed system
4	ToObtain	Domain experts	• Business rules
5	Through	Implementation	• Incremental Software Process • Phased changeover
6	Using	Design	• Modular design • Mathematical process
7	With	Modular design	• Data flow diagram
8	ByGoingTo	Block Resolution	• Senior and experienced programmers • Problem domain • Physical objects • Interactive blogs • Definition and design
9	ObtainedFrom	Deployed System	• User requirements • Users comments
10	RequirementElicitationThrough	Stakeholders and Endusers	• Stakeholders • Observation • Interview

11	TransfersKnowledgeThrough	Coding	• Document review • Brainstorming • Questionnaire • Code review • Pairwise coding

Table 3: Response from the Questionnaire

Competency Question	**Yes**	**No**
1. Ethnographic study in requirements gathering involves the study of?	66%	34%
2. What are the processes for gathering requirements from stakeholders & end-users?	71%	29%
3. What are the processes of software development?	67%	33%
4. What are the physiological processes for blocks resolution during coding?	55%	45%
5. What is the appropriate approach to software design?	70%	30%
6. What are the people and processes involved in user tasks?	66%	34%
7. What are the tools used in ethnographic study of stakeholders and end-users?	76%	24%
8. What are the stages involved in implementation?	68%	32%
9. What does a deployed system need for maintenance?	72%	28%
10. Who can business rules be obtained from in a domain?	75%	25%
11. How can low bus factor issues be handled in coding?	80%	20%
12. How can code ownership issues be resolved in coding?	74%	26%
13. How can knowledge be shared in software process?	82%	18%
14. How can knowledge be transferred in software process?	69%	31%