Published by
NIGERIAN SOCIETY OF PHYSICAL SCIENCES
Available online @ https://journal.nsps.org.ng/index.php/jnsps

J. Nig. Soc. Phys. Sci. 1 (2019) 122–130

Journal of the Nigerian Society of Physical Sciences

Original Research

Software Process Ontology: A case study of software organisations software process sub domains

R. O. Oveh*, O. Efevberha-Ogodo, F. A. Egbokhare

Department of Mathematics and Computer Science, Western Delta University, Oghara, Delta State, Nigeria

Abstract

In a domain like software process that is intensively knowledge driven, transforming intellectual knowledge by formal representation is an invaluable requirement. An improved use of this knowledge could lead to maximum payoff in software organisations which is key. The purpose of formal representation is to help organisations achieve success by modelling successful organisations. In this paper, Software process knowledge from successful organisations was harvested and formally modelled using ontology. Domain specific knowledge base ontology was produced for core software process subdomain, with its resulting software process ontology produced.

Keywords: Software Process, Software Process Ontology, Ontology, Knowledge, Formal Representation, Knowledge Representation

Article History :
Received: 06 June 2019
Received in revised form: 09 September 2019
Accepted for publication: 12 September 2019
Published: 17 December 2019

1. Introduction

Software process is a knowledge driven and knowledge intensive process that involves several other sub-processes. Software process can 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 [1-23].

Harvesting, representing and reusing knowledge within a domain leads to maximum payoff, which is desirable in most organisations [23]. Knowledge Management (KM) is defined as an effort to capture critical knowledge and share it within an organization [3, 17]. It capitalizes on the collective organizational memory to improve decision making, enhance productivity, and promote innovation [18, 19].

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 [20]. Knowledge management involves the identification and analysis of available and required knowledge [21] and 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 [22].

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 stan-

*Corresponding Author Tel. No: +2347036142579
Email address: omo_rich@yahoo.com (R.O.Oveh)

dard way of planning and organizing a software process. The major phases are requirement gathering, design and coding, implementation and maintenance. It has been identified that there are few works in literature that aim at developing ontologies covering wide portions of the SE domain, such as [4-6].

A lot of SE domain ontologies model SE subdomains [7-11]. Ref. [12] described these subdomain ontologies as weak or not interrelated, and are often applied in isolation. Thus he made an attempt to provide an integrated solution for better dealing with KM-related problems in SE by means of a Software Engineering Ontology Network (SEON). It was designed with mechanisms for easing the development and integration of SE domain ontologies, covering the main technical software engineering subdomains (i.e requirements, design, coding and testing). However, he only represented a small portion of software engineering ontology. [7], identified that the combination of ontologies of all SE subdomains would result in an ontology of the complete SE domain. He further stated that the reality is that this goal is extremely laborious, not only due to its size, but also due to the numerous problems related to ontology integration and merging, such as overlapping concepts, diverse foundational theories, and different representation and description levels, among others. He concluded that despite the challenges involved, an ontological representation covering a large extension of the SE domain remains a desired solution. This paper represents a software process ontology covering major SE subdomains (i.e. requirement gathering, design and coding, implementation and maintenance).

2. Related Literature

Ontologies have been widely recognized as a key enabling technology for KM. They are used for establishing a common conceptualization of the domain of interest to support knowledge representation, integration, storage, search and communication [2]. A domain ontology identifies the key concepts, objects and entities that exist in some knowledge domain or area of interest and the relationships between them [15, 16]. Ontologies play a significant role for knowledge sharing and as knowledge models in instructional science, technology-enhanced learning, knowledge management and training [15, 14, 13]. Ontologies consist of instances, properties and classes, where instances represent specific project data, properties represent binary relations held among software engineering concepts/instances, and classes represent the software engineering concepts interpreted as sets that contain specific project data [25]. [7], did an extensive review of SE ontologies, where he classified them into generic and specific ontology. Generic SE Ontologies, have the ambitious goal of modeling the complete SE body of knowledge; while Specific SE Ontologies, attempting to conceptualize only part (a subdomain) of this discipline.

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 [24].

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 [26].

3. Methodology

Two complementary methods were used for data collection. They are: case study and interview methods. For the case study, four (4) Software Development Organizations was studied. For reason of privacy and confidentially, the organizations studied are not referenced by their names. Table 1 shows the details of the organisations.

From Table 4, the domain concept, the data value/property and the instances are specified. For example the entity domain expert in number 9 has the property: name that can take a data value string, data property domain that can take a data value string and a data property years of experience that can take a data property integer. It also has instances of the class as: business rules and directory of experts. Figure 6 shows the domain concepts and their instances

4. Result

Four main subdomains were identified as knowledge entities in a typical software development process irrespective of the life cycle model adopted: Requirements Definition, Design & Coding, implementation and maintenance. These are core human centric activities performed by developers that create opportunities for sharing tacit knowledge during the software development process. This research used both inductive and deductive analysis by first identifying keywords related to software development process and then grouping the keywords into categories related to requirements definition, coding, implementation and maintenance. For each process activity, a Union of the useful themes obtained each case study was used to determine the useful knowledge constructs for that activity. That is, for each Process Activity (PA), Knowledge Harvested (KH) for the software process is given as

$$\text{KH(PA)=Case 1(PA)} \bigcup \text{Case 2 (PA)} \bigcup \text{Case 3 (PA)} \bigcup \text{Case 4 (PA)} \qquad (1)$$

5. Software Process Ontology

There is no globally accepted methodology for Ontology construction [28] but the development is usually an iterative process. A 5-step iterative process from [29, 30] was adopted for the Ontology construction:

Table 1: Details of the organisations

Case Study	Years of existence	Software Area
1	15	Human resources systems, Payroll systems, and portal for schools.
2	22	Banking solutions, Human Resource management systems, Process Control Systems, Payroll Systems, Security systems, Materials management systems
3	10	security solutions, and Web Development
4	12	Human resource Management Systems, Third Party Online Payment Solutions, Payroll and other financial systems

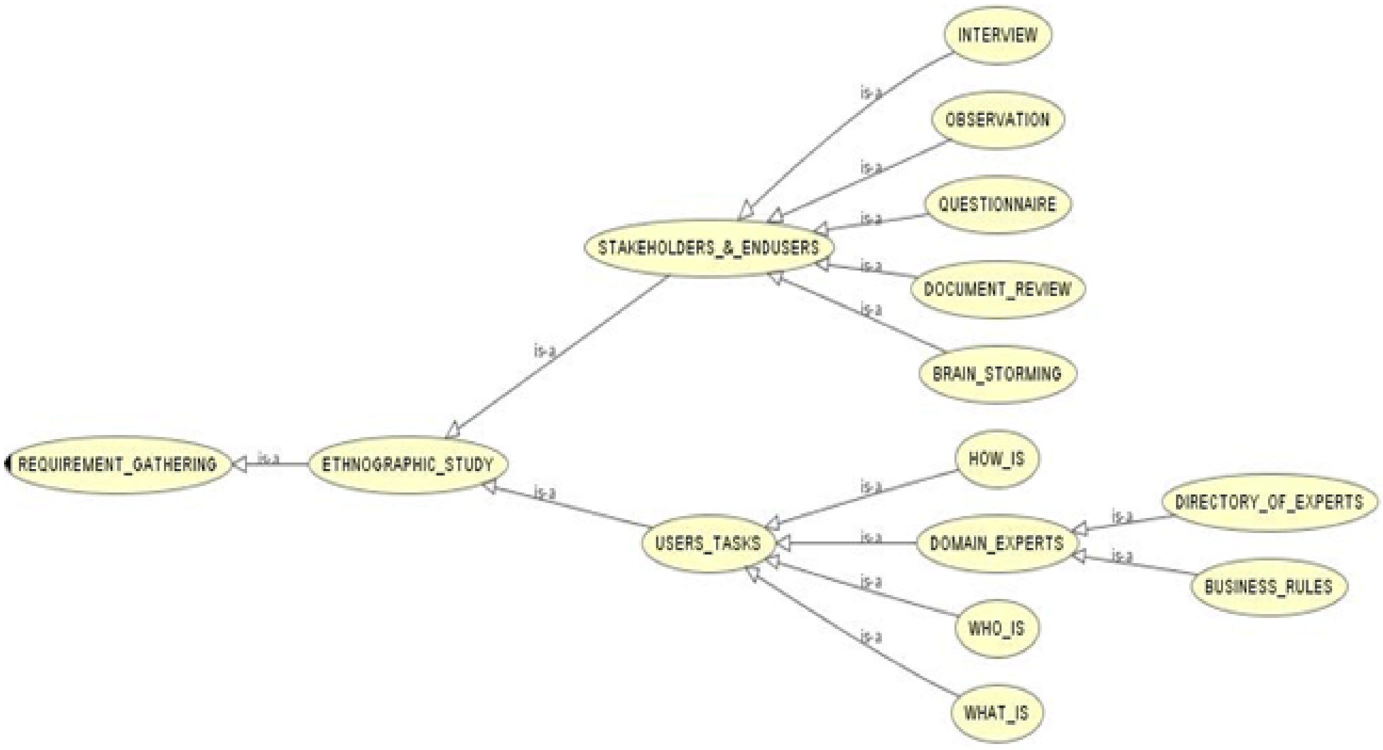

Figure 1: Requirement Definition Sub-domain Ontology Visualization

- **Step 1: Identify the key concepts of the domain**
  The first step in the Ontology building process is to identify the concepts and map them into the relevant knowledge entities. To provide clarity and efficiency in the formal representation of software process knowledge, the concepts from the harvested knowledge were categorized into the four main knowledge entities:

  1. Knowledge-Creation
  2. Knowledge-transfer
  3. Knowledge-sharing
  4. Knowledge-documentation.

  Each concept (Table 2) connotes a software process activity that describes a task, function, action, strategy, or reasoning process. A Concept is a collection of objects. It is the fundamental element of a domain and usually represents a group or class whose members share common properties [30]

- **Step 2: Organisation of the concepts into a hierarchy**
  The purpose of this categorization is to establish a systematic relationship between the knowledge entities and the specific software development process activities. Competency questions was used here in creating the ontology. Competency questions as defined by some methodologies for ontology engineering describe what kind of knowledge the resulting ontology is supposed to answer. According to [27] one of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge based on the ontology should be able to answer. The following competency questions were used:

  1. Ethnographic study in requirements gathering involves the study of?
  2. What are the processes for gathering requirements from stakeholders and end-users?
  3. What are the processes of software development?
  4. What are the physiological processes for blocks resolution during coding?

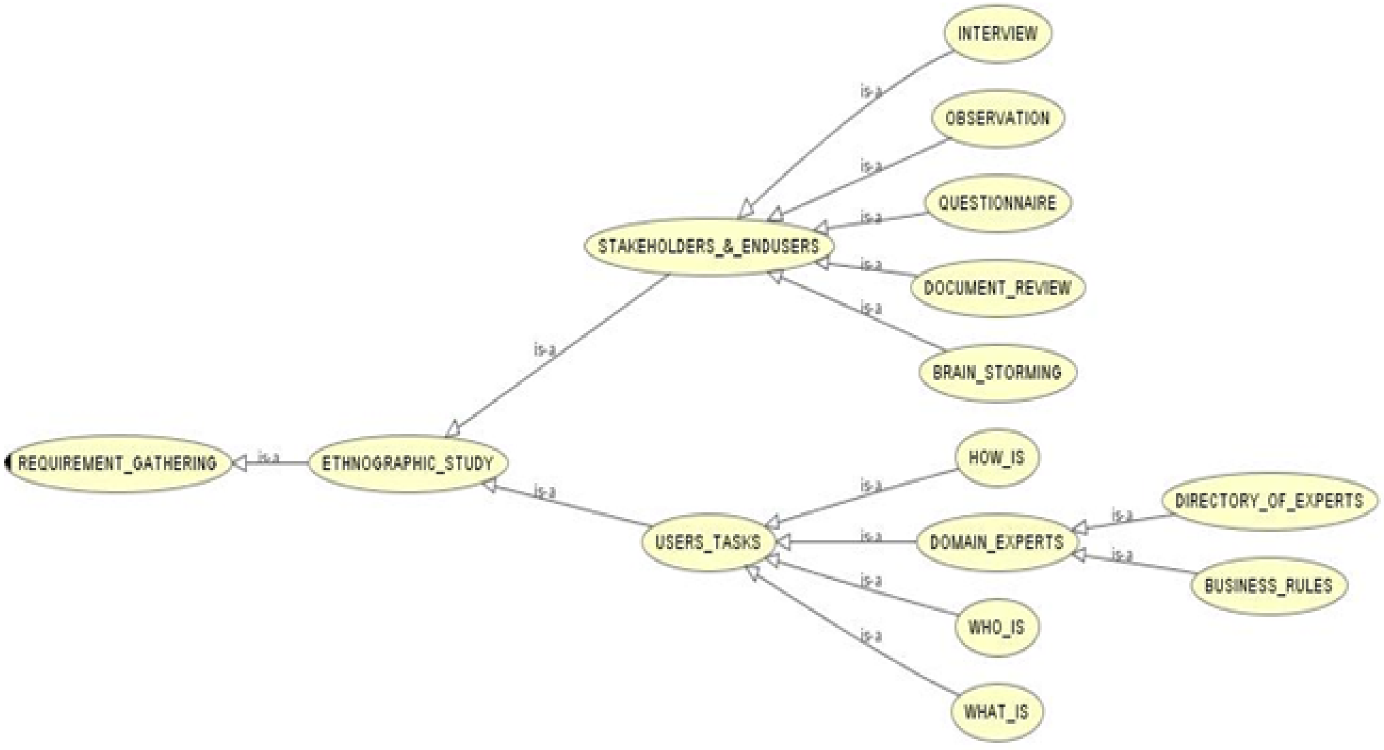

Figure 2: Design and Coding Sub-domain Ontology Visualization

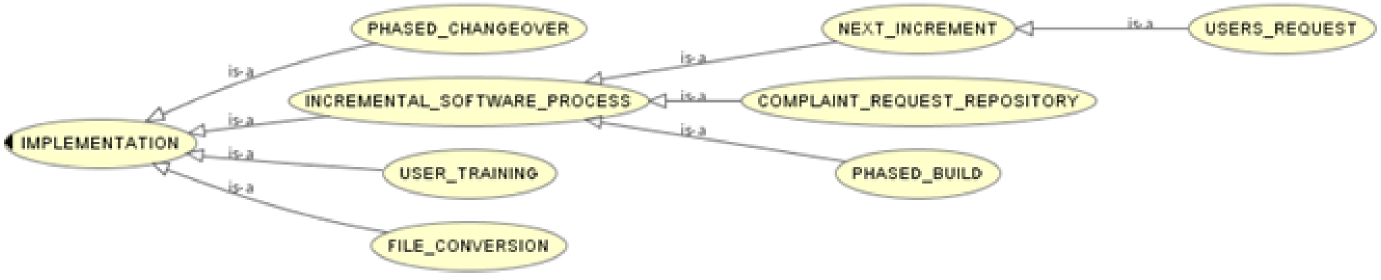

Figure 3: Implementation Sub-domain Ontology Visualization

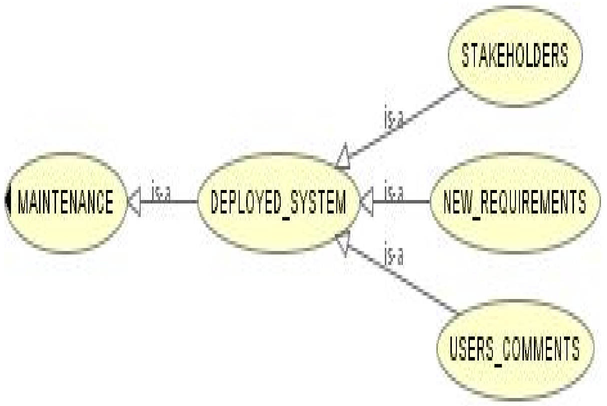

Figure 4: Maintenance Sub-domain Ontology Visualization

5. What are 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?

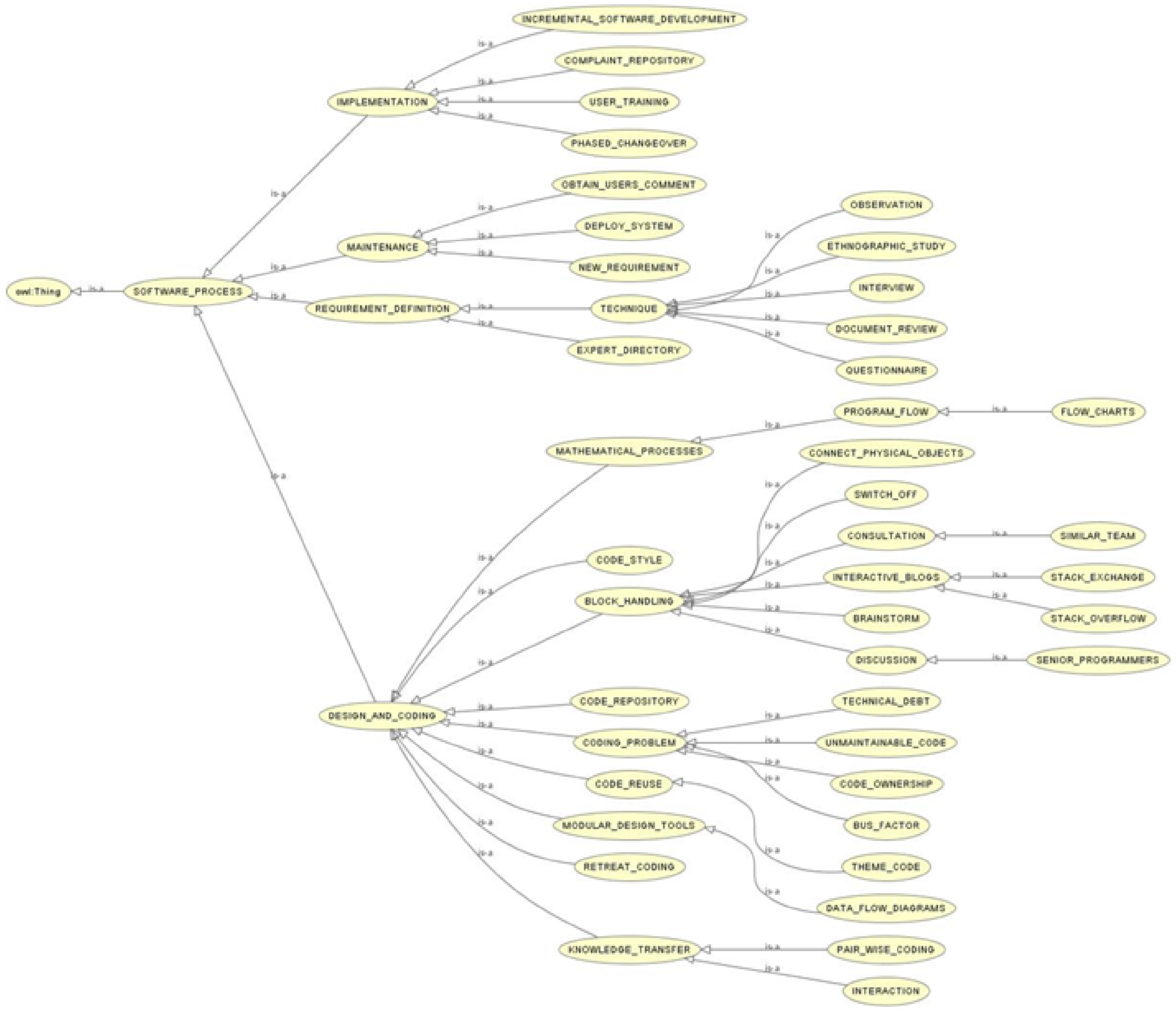

Figure 5: Software Process Ontology Visualization

- **Step 3: Determine the properties of each class**
  Every objects have both data property and object property. The object property shows the relationship between classes or instances, and the data property shows the relationship between instances and the data value. It provides a logical relationship between objects. Tables 3 and 4 present sample Object and Data property defined for some objects in the software process ontology.

- **Step 4: Add logical expressions**
  Axioms represent assertions formulated in a logical form that together comprise the core knowledge that the ontology describes in its domain of application. They are used to model sentences that are always true. They provide a powerful way to add logical expressions to ontology and they are used to verify the consistency of the ontology. Axioms are usually added by default in protege.

- Step 5: Create the ontology
  Protégé 5.0 was then used to build the software process knowledge ontology, with its subdomains.as shown in Figure 1-5

6. Discussion

The requirements definition (figure 1) is the first stage in software development process. It is a very critical phase. A major finding from the organizations studied is that Ethnography study was used to obtain the software requirements. It involves studying the organization's culture to understand the key elements and observable patterns of behaviour. This usually requires interaction between stakeholders from the software de-

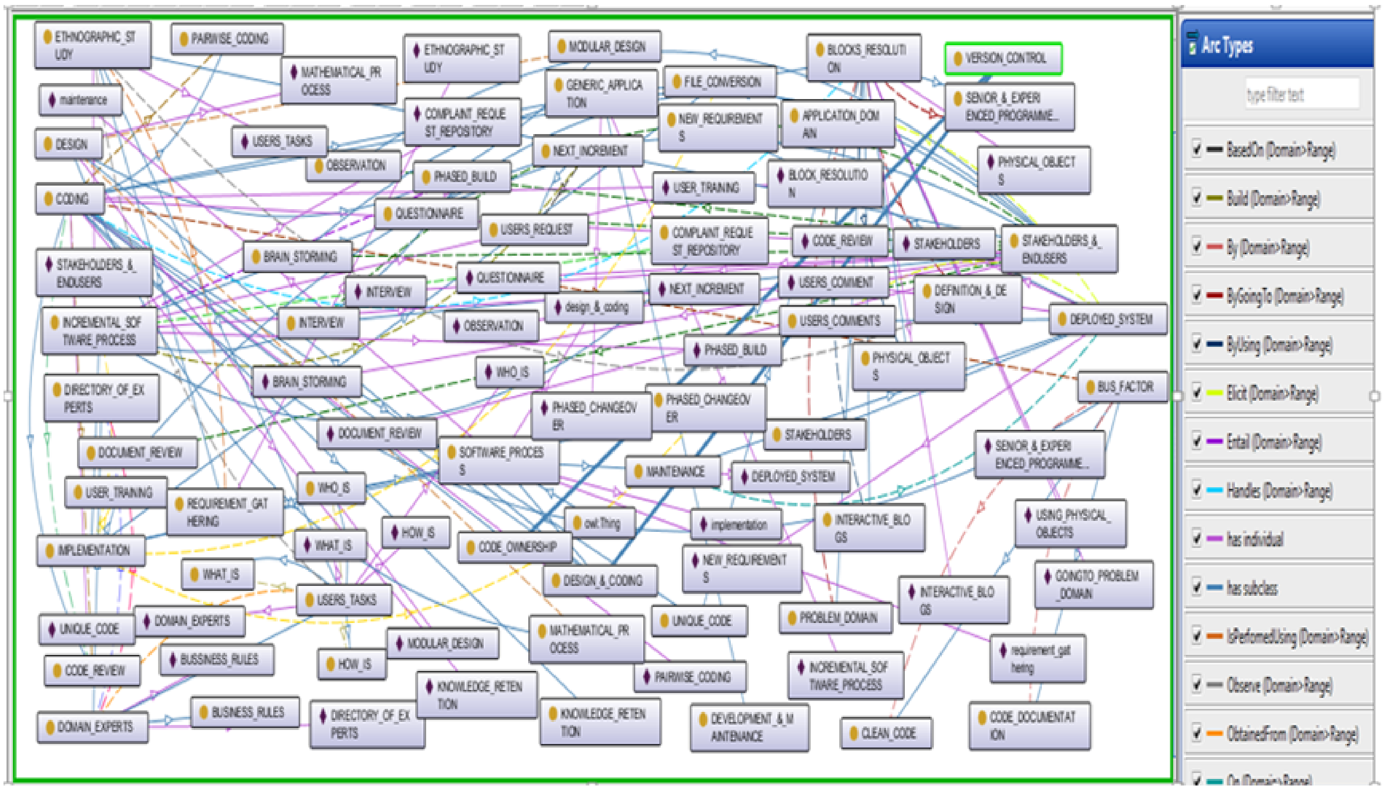

Figure 6: Software Process Ontology showing concept, instances and their relationship

velopment organization and those from the user organization in order to gain a better understanding of the problem. It was observed from the study that quality time is dedicated to identifying and interacting with the end users during requirements elicitation. These series of interactions provide opportunity for the company to distinguish itself and to learn about how the users' tasks are performed. Users are asked to discuss their routine tasks with the requirements elicitation team and they are informed about the services to be provided by the proposed system. This helps to prevent user resistance to the new software system and provides opportunity for users to discuss their daily routines with the developers in an informal way. During the process, apart from using informal interviews to gather data, participant observation, questionnaires, documents review and other suitable requirements elicitation techniques are also used when necessary. Sometimes, the user's expressions and reactions send useful signals regarding what they expect in the system. One of the organizations studied adopted the Agile software Method were the initial requirements are quickly developed into the first build which is installed for the users and additional features are reported as feedbacks for the next build. Users e-mail addresses and phone numbers are collected. A repository is created to store user complains and any additional features requested.

Figure 2 shows the approach to design and coding. For design it should be broken down into mathematical process and made modular. Approaches to coding should include: pairwise coding, avoiding code ownership, code review, and encourage knowledge retention. Bus factor issue should be resolved by creating clean code and documenting codes. Blocks which are dead ends during coding should be resolved by going to: senior and experienced programmers, problem domain, back to definition and design, physical objects like games, and interactive blogs like stack overflow or stack exchange.

Implementation in figure 3 should adopt phased change over approach instead of a holistic approach. It should be done incrementally. Maintenance in figure 4 requires the system to be first deployed, and further requirements obtained from stakeholders and end users based on their usage. Figure 5 is the various subdomain put together to form the software process ontology.

7. Conclusion

Software process knowledge is a knowledge driven process with sub-processes. This knowledge is latent and could be lost if not formally harvested and documented. An improved use of this knowledge could lead to maximum payoff in software organisations. This is the heart of knowledge management, which focuses on knowledge capturing and sharing. This paper presents a generic software process domain ontology, covering the main technical software engineering subdomains of requirements definition, design & coding, implementation and maintenance.

Acknowledgments

We thank the referees for the positive enlightening comments and suggestions, which have greatly helped us in making

Table 2: Knowledge Entities and software process Concepts

Knowledge Entities	Software Process Concept
Knowledge Creation	Ethnographic study
	Stakeholders
	-New requirement
	- Users comment
	Exit interviews
	Domain experts
	-Business rules
	Questionnaire
	Document Review
	Block resolution
	Users Request
Knowledge Transfer	Knowledge Retention
	Business rules
	User Training
	Mentoring
	Pair wise coding
	Code review
	Interactive blogs
Knowledge Sharing	Experienced programmers
	Team interaction
	Bus factor
	User tasks
	Code ownership
	Version control
	What-is
	Who-is
	How-is
Knowledge Documentation	Directory of experts
	Requirements document
	Code repository

improvements to this paper.

References

[1] I. Rus and M. Lindvall, "Knowledge Management in Software Engineering" IEEE Software **19** (2002) 26.

[2] D. O'Leary, "Using AI in knowledge management: knowledge bases and ontologies", IEEE Intelligent Systems, **13** (1998) 34.

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

[4] O. Mendes and A. Abran, *Issues in the Development of an Ontology for an Emerging Engineering Discipline. First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Eng. (ONTOSE)*, Alcala Henares, Spain (2005)

[5] M. A. Sicilia, J. J. Cuadrado, E. Garcia, D. Rodriguez and J.R. Hilera, "The Evaluation of Ontological Representation of the SWEBOK as a Revision Tool. In: 29th Int. Computer Software and Application Conference (COMPSAC)", Edinburgh, UK. (2005) 26

[6] P. Wongthongtham, E. Chang, T. Dillon, and I. Sommerville, "Development of a Software Engineering Ontology for Multisite Software Development. IEEE Transactions on Knowledge and Data Engineering", **21** (2009) 1205

[7] C. Calero, F. Ruiz and M. Piattini, "Ontologies for Software Engineering and Soft-ware Technology", Springer Science & Business Media (2006)

[8] E.F. Souza, R.A.Falbo and N.L.Vijaykumar, "Using Ontology Patterns for Building a Ref-erence Software Testing Ontology. In: 17th IEEE Int. Enterprise Distributed Object Computing Conference Workshops (EDOCW)", Vancouver (2013) 21.

[9] C. Gonzalez-Perez and B. Henderson-Sellers, "An Ontology for Software Development Methodologies and Endeavour", (2006). In: [7] A.C.Bringuente, R.A. Falbo and G. Guizzardi, "Using a Foundational Ontology for Reengineering a Software Process Ontology. Journal of Information and Data Management", **2** (2011) 511.

[10] R. F. Calhau and R.A. Falbo, "An Ontology-based Approach for Semantic Integration. In: 14th IEEE International Enterprise Distributed Object Computing Conference, Vitória, Brazil. Los Alamitos: IEEE Computer Society", (2010) 111.

[11] F. Borges Ruy, R. de Almeida Falbo, M. Perini Barcellos, S. Dornelas Costa and G. Guizzardi "SEON: A Software Engineering Ontology Network" (2016). In: E. Blomqvist , P. Ciancarini, F. Poggi and F. Vitali (eds) "Knowledge Engineering and Knowledge Management. EKAW 2016. Lecture Notes in Computer Science", Springer, Cham (2016) 10024.

[12] A. Zouaq and R. Nkambou, "Evaluating the generation of domain ontologies in the knowledge puzzle project. IEEE Transactions on Knowledge and Data Engineering", **21** (2009) 1559. http://dx.doi.org/10.1109/TKDE.2009.25

[13] M. D. Kickmeier-Rust and D. Albert, "The ELEKTRA ontology model: A learner-centered approach to resource description. Advances in Web Based Learning - ICWL 2007", Lecture Notes in Computer Science. Berlin: Springer **4823** (2008) 78.

[14] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?", International Journal of Human-Computer Studies **43** (1995) 907. *http://dx.doi.org/10.1006/ijhc.1995.1081*

[15] A. Zouaq and R. Nkambou, Building domain ontologies from text for educational purposes. IEEE Transactions on Learning Technologies **1** (2008) 49 http://dx.doi.org/10.1109/TLT.2008.12

[16] M. Alavi and D. E. Leidner, "Knowledge management and knowledge management systems: Conceptual foundations and research issues", MIS Quarterly, **25** (2001) 107.

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

[18] E. Perez, "Knowledge Management in the Library-Not. Database Magazine" **22** (1999) 75.

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

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

[21] A. P. J. Abdul-Kalam, "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 (2004)

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

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

[24] P. Wongthongtham, N. Kasisopha, E. Chang and T. Dillon, "A Software Engineering Ontology as Software Engineering Knowledge Representation", International Conference on Convergence and Hybrid Information Technology. IEEE. (2008) 668

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

[26] M. Gruninger and M. S. Fox, "Methodology for the design and evaluation of ontologies", In IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal (1995)

[27] N. F. Noy and D. L. McGuiness, "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 (2001).

[28] B. Kapoor and S. Sharma, "A Comparative Study Ontology Building Tools for Semantic Web Applications", International Journal of Web and Semantic Technology, **1** (2010) 1.

[29] A. F. Sawsaa, *Ontological Engineering Approach of Developing*

Table 3: Object Property

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, User requirements
9	ObtainedFrom	Deployed System	Users comments, Stakeholders
10	RequirementElicitationThrough	Stakeholders and Endusers	Observation, Interview, Document review, Brainstorming, Questionnaire
11	TransfersKnowledgeThrough	Coding	Code review, Pairwise coding

Table 4: Data Property

S/N	Domain Concept	Property/Data Value	Instances
1	Ethnographic study	Purpose(string), target(string), finding(string)	Stakeholders and EndUsers, User tasks
2	Coding	Title (String), Phase(string)	Code Review, Knowledge Retention, Generic application, Unique code, Pairwise coding, Block resolution
3	Design	Name(string), Model (string)	Modular design, Mathematical Process
4	Users tasks	Name(string), specification(string)	Domain experts, Who is, What is, How is
5	Maintenance	Date(date), details(string)	Deployed System
6	Implementation	Title(string), purpose(string), finding(string) Name(string)	User Training, File conversion, Incremental Software Process, Phased changeover
7	Deployed system	(integer), date(date)	New Requirement, Users comments
8	Incremental Software Process	process no(string), date(date)	Next increment, Phased build, Complaint request repository
9	Domain Experts	Name(string), domain(string), years of experience (integer)	Business rules, Directory of experts
10	Stakeholders and end users	Name(string), organisation (string), portfolio (string), domain (string)	Observation, Interview
11	Block resolution	Definition and design (string), problem domain (string), experience programmers (string), interactive blogs (string), physical objects (string)	Document review, brainstorming, senior and experienced programmers, problem domain, physical objects, interactive blogs, definition and design

Ontology of Information Science, Anchor Academic Publishing. https://books.google.com.ng/books?isbn=3954894483 (2015)

[30] O. Corcho and A. Gomez-Perez, "A Roadmap to Ontology Specification Languages", in Rose Dieng and Olivier Corby (eds.), Knowledge

Engineering and Knowledge Management. Methods, Models and Tools, Springer, Berlin, (2000) 80.