International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 08, August-2019

Analyse the Convergence of Multi-Domain Research Opportunities in Software Engineering

Dr. A. Saranya
Assistant Professor
Department of Computer Application
V.V.V College for Women Virudunagar

Dr. A. Sumithra
Asistant Professor
School of Computer Science & Engineering
Sriramakrishna Institute of Technology

***Abstract*:- Software Engineering (SE) is taking part in an important role in the software industry as a result of specific software is required in virtually every industry, in each business, and for each operate. There are completely different levels are available in software engineering and they serve several functions throughout the application lifecycle. At the early stage, the industry developed an application mostly for personal use and small scale industries. In that time, the SDLC processes are very marginal and moderated, and even the accounting data is also moderated. The revolution of software industry demands the amount of multifaceted applications and obviously, it leads to the high quality of SDLC. The human-driven era of software development meant writing rule-based code that solved deterministic problems using logic however it needs a transition to the requirement of large scale software developments. It's going to be a replacement proposal of SDLC or cross-industry standards like data mining, Machine Learning, etc. This paper argues the future opportunities and challenges of software Engineering and highlights how actually the SE wants cross-industry standards to improve the quality of large scale software developments.**

Keywords:* *Software Engineering, AI, Natural Language Processing, Data Mining, Big data

1. INTRODUCTION

The direction of contemporary research is unpredictable notably in the field of SE [1]. SE grows on a daily basis and it moves around with a completely different dimension. The demand for modern software development forum is extremely high and therefore the SE ought to gratify that. Even the core processing of SDLC isn't able to cope-up the industry needs. The convergence of technologies makes SE successful and deterministic. The data acquiring and processing technologies like data mining, big data, AI and NLP, etc. are often used for SE to provide constructive and effective results within the time.

This paper highlights the various research dimensions of SE and the way it will incorporate with different interdisciplinary techniques. The organization of the paper as follows: Section II discusses the summary and the basic parts of software engineering. Section III point-out the important researches contributions already created by the researchers. Section IV describes the convergence of SE and data mining. This section in an elaborate way mentioned how the data mining concepts are really contributed to SDLC once the developments of large scale applications. how the AI contributes on software development for rapid and cost-effective SDLC is discussed in Section V. AI is going to be the one of the important convergence of SE within few years and it's through the huge number of research problem in SE. Section VI highlights the contribution of NLP (Natural Language Processing) in multilingual software developments and text extraction in SE. Ontologies and its contributions towards the SE are discussed in Section VII. Section VIII discusses how big data contributes to extracting knowledge from the huge variety of SDLC data. Section IX concludes the paper.

2. BITS AND PIECES OF SOFTWARE ENGINEERING

Software engineering has evolved over the last fifty years (the term coming back into widespread use after it had been utilized in the title of a conference on the subject sponsored by North Atlantic Treaty Organization in 1968), at first as a response to the alleged software crisis (the issues that organizations had to produce quality software systems on time and on budget) of the 1960s and 1970s.

Software engineering (SE) has been outlined as "the application of a systematic, disciplined, quantitative approach to the development, operation, and maintenance of software; that's, the application of engineering to software" [2]. Because the field has matured, SE has developed a variety of approaches to areas like software requirements, software design, software testing, and software maintenance. Software development processes like the waterfall model, incremental development, and also the spiral model are with success applied to produce high-quality software on time and beneath budget [3]. More recently, agile software development has gained popularity as another to the additional traditional development ways for the development of advanced systems [4].

3. LITERATURE REVIEW

Taylor et al. [5] have produced a comprehensive survey covering the most recent applications of data mining to software engineering. They also discuss the issues one might encounter in mining software data and the necessary conditions for success. Halkidi et al. [6] have written one of the most thorough papers on the subject. Their work features in depths look at the data mining techniques and how they can be effectively applied in software engineering.

Aouf et al. [7] described how clustering techniques could be used to discover hidden patterns in data to gather valuable information. Chang and Chu [8] showed how Association rule mining could be used to detect software defects. Gegick et al. [9] demonstrated the importance and usage of text mining in bug identification whereas Runeson et al. [10] showcased the capabilities of NLP in tackling duplicate defect reports. Islam and Brankovic [11] proposed techniques to ensure privacy in data mining. This was mainly done by filling parts of the dataset with noisy data. On the other hand, Ma and Chan [12] in their work suggested iterative mining for mining overlapping patterns in noisy data. While, Islam and Brankovic [11] were concerned with preserving privacy with the help of noisy data, Ma and Chan [12] dealt with the elimination of noisy data to achieve the objective of extracting valuable information.

Natural Language Text is understood by humans with little effort. The importance of textual processing on natural language text is discussed by Viliam [13]. Farid discusses the use of UML's class diagram in generation of natural language text. The paper describes various NL based systems to strengthen the view point of generating NL specification from class diagrams. The paper shows use of WordNet to clarify the structure of UML string names and generating the semantically sound sentences [14]. Reynaldo uses controlled NL text of requirements to generate class models. The paper describes some initial results arising out of parsing the text for ambiguity. The paper introduces a research plan of the author to integrate requirement validation with RAVEN project [15].

Deva Kumar, et al., created an automated tool (UMGAR) to generate UML's analysis and design models from natural language text. They have used Stanford parser, Word Net 2.1 and Java RAP to accomplish this task [16]. Sascha, et al., proposed a round trip engineering process by creating SPIDER tool. The paper addressed the concerns about errors at requirement level being propagated to design and coding stages. The behavioural properties shown from the NL text are utilized to give developer a UML model [17].

Researchers have so far proposed many different synergies between software engineering and ontologies. For example, ontologies are proposed to be used in requirement engineering [24], software modeling [25], model transformations [26], software maintenance [27], software comprehension [28], software methodologies [29], and software community of practice [30]. Moreover, software engineering technologies are proposed for modeling and reasoning over ontologies. These synergies between ontologies and software engineering have also attracted attention of standardization bodies and have some on-going activities. Ontology-Driven Architecture (ODA) is an effort of the W3C's Software Engineering Best Practices Working Group that tries to develop best practices for using ontologies in software engineering [31]. Probably, the most important result so far is the Ontology Definition Metamodel (ODM) that is proposed to be the Object Management Group (OMG)'s standard [32]. The ODM standard allows for integrating ontology languages (i.e., ontologies) into the software development process based on model-driven engineering principles [33]. Although all of these different efforts demonstrate many benefits to different aspects of software and ontology engineering or give a nice description of the state of the art in the area [29, 33], none of them analyze and evaluate applications of ontologies in different aspects of software engineering by following a comprehensive software lifecycle framework.

4. DATA MINING AND SOFTWARE ENGINEERING

Due to its capability to handle massive volumes of data and its efficiency to spot hidden patterns of knowledge, data mining has been projected in research works as mean to support industrial-scale software system maintenance, debugging, testing (Table 1). The mining results will facilitate software system engineers to predict software failures, extract and classify common bugs, confirm relations among categories in libraries, analyse defect data, discover reused patterns in source code and thus modify the development procedure. In general, terms, victimization data mining practitioners and researchers can explore the potential of software system engineering data and use the mining results in order to better manage their projects and to produce higher quality software systems that are delivered on time and on budget. In essence, data mining for software system engineering is rotten on 3 axes [18]: the goal, the input data used, and so the mining technique used. For example, the goal is additionally to enhance code completion systems [19].

Table 1. Development stages with mining techniques

Software Development Stage	**Data Mining Techniques**	**Input Data**	**Data Analysis Result**
Requirement Elicitation	Classification	Documentation	Classification of requirements
	Text mining	Mailing lists	Data Summarization
Design	Clustering	Design document	Data gathering, labeling
Implementation	Clustering	Source code	Software processes
	Classification	SCM	Bug tracking
	Text	Mining Source code	Bug tracking
	Frequent Pattern Mining & Association rules	Defect	Defect Correction
		Program dependence graph	Neglected Conditions
Testing	Classification	I/O variables of software system	A network producing sets for function testing
		Program executions	Software behavior classifiers
	Clustering	Execution Profiles	Clusters of execution profiles

5. ARTIFICIAL INTELLIGENCE AND SOFTWARE ENGINEERING

Software engineering has seen a vast transformation over the past few years. AI and software intelligence tools aim to create software development easier and a lot of reliable. In step with a Forrester research report [20] on AI's impact on software development, machine-driven testing and bug detection tools use AI the foremost to enhance software development (Fig 1).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 08, August-2019

The age of machine learning is here and AI for software development can forever amendment programming. It’s no longer regarding process if-then-else cycles, and it's become a lot of concerning selecting the proper information to coach the neural network which is able to solve the given downside without human intervention. This can be a revolution in the manner issues are solved, the tools used, the mental attitude and even the definition of what a developer will. We are going to explore a number of the ways in which AI will enhance software development, some pitfalls and at last, why this approach is effective.

Rapid Prototyping - Turning business needs into technology products usually need months if not years of designing, however machine learning is shortening this process by facultative less technical domain consultants to develop technologies victimization either linguistic communication or visual interfaces.

Intelligent Programming Assistants - Developers pay the overwhelming majority of their time reading the documentation and debugging code. Sensible programming assistants will cut back this time by providing just-in-time support and suggestions, like relevant document, best practices, and code examples. Samples of such help embody Kite for Python and Codota for Java.

Automatic Analytics & Error Handling - Programming assistants may learn from past expertise to spot common errors and flag them automatically throughout the development part. Once a technology has been deployed, machine learning may be accustomed analyse system logs to quickly and even proactively flag errors. In the future, it'd even be attainable to alter the software system to alter dynamically in response to errors without human intervention.

Automatic Code Refactoring - Clean code is crucial for team collaboration and long-run maintenance. As enterprises upgrade their technologies, large-scale refactoring is inevitable and sometimes painful requirements. Machine learning will be accustomed analyse code and mechanically optimize it for interpretability and performance.

Precise Estimates - Software development notoriously goes over budget and over timelines. Reliable estimates need deep experience, understanding of context, and familiarity with the implementation team. Machine learning will train on data from past projects - like user stories, feature definitions, estimates, and actuals - to predict effort and budget additional accurately

Strategic Decision-Making - A significant portion of your time is spent debating that product and options to grade and which to cut. An AI answer trained on each past development projects and business factors will assess the performance of existing applications and facilitate both business leaders and engineering groups determine efforts that may maximize impact and minimize risk.

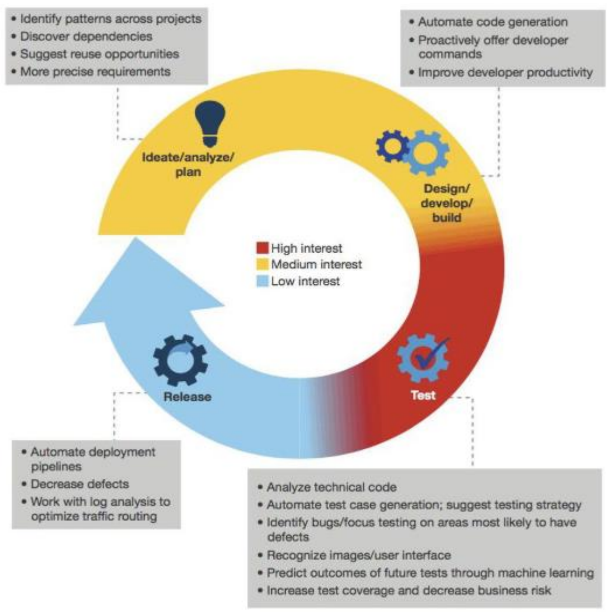
• Identify patterns across projects
• Discover dependencies
• Suggest reuse opportunities
• More precise requirements
• Automate code generation
• Proactively offer developer commands
• Improve developer productivity
Ideate/analyze/ plan
Design/ develop/ build
High interest
Medium interest
Low interest
Release
Test
• Automate deployment pipelines
• Decrease defects
• Work with log analysis to optimize traffic routing
• Analyze technical code
• Automate test case generation; suggest testing strategy
• Identify bugs/focus testing on areas most likely to have defects
• Recognize images/user interface
• Predict outcomes of future tests through machine learning
• Increase test coverage and decrease business risk

Fig 1. AI for SDLC

6. NLP AND SOFTWARE ENGINEERING

Software Development Life Cycle consists of a group of phases. These phases offer guidelines to develop software package. Natural language processing may be applied in each section concerned inside the Software Development Life Cycle. Its additional precise exploitation it once the artefacts of part or activity are plain text. Plain text may be used as input for natural language process tasks. Basically, all the activities during which the humans interpret the document there's the scope of textual generation.

The supposition, inexplicit several NLP-based requirements capture proposals [21], that each one requirements for a future system unremarkably exist in textual type, isn't borne out in reality. It’s true that some data happens naturally as text, generally process descriptions or predefined procedures, but far more is to be found in diagrams or within the physical reality encompassing the client. To rely alone on the text as a source of information or to expect the client to cut back all his or her demands to a textual type is clearly unworkable. Assuming but that the requirements definition task is being performed by an intelligent human who a considerable body of the machine-readable text is obtainable, there's no doubt that tools to scan, search, browse and tag that text might assist in developing a full and correct statement of desires [22]. This might positively not imply the automated understanding of a free text.

7. ONTOLOGIES FOR SE

Ontologies are used for formal illustration of domain knowledge. Knowledge-based Applications use Ontologies for knowledge sharing that forms the primary use of Ontologies. Ontology development is essentially aimed toward AI specialists with knowledge of various techniques originating from the sector of AI. This knowledge is however unknown to an outsized section of the software industry. To bridge the gap between knowledge of software Engineering practitioners and AI techniques, many proposals are created suggesting the utilization of Ontologies within the field of software Engineering.

Proposals embrace the utilization of UML diagrams within the development of Ontologies. In fact, the recipient software developed by university incorporates a tab that unambiguously defines the utilization of UML diagrams in ontology development. This practicality will be used to develop a UML diagram from the ontology developed.
However, it also can be seen, that software engineering approaches themselves don't change the representation of ontology ideas derived from description logics and different concepts enclosed within the semantic web Languages.

8. BIG DATA FOR SE

As datasets handled by software package are perpetually increasing, aside from supply novel algorithms, new system architectures and software infrastructures ready to address the 5Vs of big data [23], it's time for software itself to profit from the intelligence extracted from massive sets of data like software source code, commits and forks, bugs, warnings and notifications, problems from backtracking systems, logs of any kind, commits, demographics, coding patterns, requirements, user behaviours, user profiles, etc. research challenges for software engineering during this direction embody novel tools using techniques of machine learning and data mining to reveal hidden knowledge aspects and extract information from sensor-based architectures, excavating knowledge that is not possible for humans to dig out, however is necessary to be brought into human attention and feeling for improving software qualities, learning the evolution / termination of application frameworks, open source components, analysis of user trends and preferences and behavior with systems to higher understanding users' wants, tools and strategies for distinctive feature and performance improvement opportunities, distinctive root causes of failures and system halts based on log files (massively big >>GBs) or lightning-fast updating) returning from numerous complicated distributed systems and infrastructures10, insights collected at runtime on symptoms and context changes triggering adaptations, and perform prophetic and prescriptive analytics for proactive planning and preparation of adaptation actions.

9. CONCLUSION

This paper elaborately argues that the Software Engineering will be collaborating with other domain to produce better software and applications. The increasing of software development data demands new text processing methods. The data mining, big data are the technologies will compensate the demand for text processing. AI and NLP is the domain to give support to SE to develop error-free and rapid application developments. This paper outlines the research collaborations and challenges of those domains and future directions.

REFERENCE

[1] Xu, Haiping, Future Research Directions of Software Engineering and Knowledge Engineering. International Journal of Software Engineering and Knowledge Engineering. 415-421, 2015.
[2] Abran, A., Moore, J.W., Bourque, P., Dupuis, R., Tripp, L.L.: Guide to the Software Engineering Body of Knowledge, IEEE, 2004
[3] Bhalerao, Shilpa & Puntambekar, Devendra & Ingle, Maya, Generalizing Agile Software Development Life Cycle. International Journal on Computer Science and Engineering. 1, . 2009.
[4] Agile Manifesto, http://agilemanifesto.org
[5] Taylor, Q.and Giraud-Carrier, C. "Applications of data mining in software engineering", International Journal of Data Analysis Techniques and Strategies, Volume 02, Issue 03, Page No (243-257), July 2010.
[6] M. Halkidia, D. Spinellisb, G. Tsatsaronisc and M.Vazirgiannis, "Data mining in software engineering", Intelligent Data Analysis 15, Page No (413–441), 2011.
[7] M. Aouf, L. Lyanage, and S. Hansen, "Critical review of data mining techniques for gene expression analysis," International Conference on Information and Automation for Sustainability (ICIAFS) 2008, Page No (367-371), 2008.
[8] C. CHANG and C. CHU, "Software Defect Prediction Using Inter transaction Association Rule Mining", International Journal of Software Engineering and Knowledge Engineering, Volume 19, Issue 06, Page No (747-764), September 2009.
[9] M. Gegick, P. Rotella and T. Xie, "Identifying security bug reports via text mining: an industrial case study", Mining Software Repositories (MSR), 7th IEEE Working Conference, Page No (11 – 20), 2010.
[10] P. Runeson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing", Software Engineering, 2007. ICSE 2007. 29th International Conference, Page No (499 – 510), 2007.
[11] M. Z. Islam and L. Brankovic, "Detective: a decision tree based categorical value clustering and perturbation technique for preserving privacy in data mining," Third IEEE Conference on Industrial Informatics (INDIN), Page No (701-708), 2005.
[12] P. C. H. Ma and K. C. C. Chan, "An iterative data mining approach for mining overlapping coexpression patterns in noisy gene expression data," IEEE Trans. NanoBioscience, Volume 08, Issue 03, Page No (252-258), September 2009.
[13] V. Simko, P. Kroha and P. Hnetynka, "Implemented domain model generation", Technical Report, Department of Distributed and Dependable Systems, Report No. D3S-TR-2013-03, 2012.
[14] F. Meziane, N. Athanasakis and S. Ananiadou, "Generating Natural Language Specifications from UML Class diagrams", Requirement Engineering Journal, Springer-Verlag, London, vol. 13, no. 1, pp. 1-18, 2013.
[15] R. Giganto, "Generating Class Models through Controlled Requirements", New Zealand Computer Science Research Conference (NZCSRSC), Christchurch, New Zealand, 2008.
[16] G. Lu, P. Huang, L. He, C. Cu and X. Li, "A New Semantic Similarity Measuring Method Based on Web Search Engines", WSEAS Transaction on Computer, ISSN: 1109-2750, vol. 9, Issue 1, 2010.
[17] S. Konrad and B. H. C. Cheng, "Automated Analysis of Natural Language Properties for UML Models", [Online available], 2010.
[18] T. Xie, S. Thummalapenta, D. Lo and C. Liu, Data mining for software engineering, Computer 42, 55–62, 2009.
[19] Marcel Bruch, Martin Monperrus, Mira Mezini. Learning from Examples to Improve Code Completion Systems. In Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM Symposium on the Foundations of Software Engineering, ACM, 2009.
[20] https://reprints.forrester.com/#/assets/2/108/'RES121339'/reports
[21] M Saeki, H Horai, H Enomoto, "Software Development Process from Natural Language Specification", 1lth International Conference on Software Engineering, 1989.
[22] P Loucopoulos P & R E M Champion, "Concept Acquiisition and Analysis for Requirements Acquisition", IEE Software Engineering Journal, (2) 1990.
[23] Kalbandi, Ishwarappa & Anuradha, J, A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology. Procedia Computer Science, 2015.
[24] Seok Won Lee and Robin A. Gandhi. Ontology-based Active Requirements Engineering Framework. In Proc. of the 12th Asia-Pacific Software Eng. Conf. 481–490, 2005.
[25] Holger Knublauch. Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with

Protege/OWL. In Proc. of 1st Int'l WSh on the Model-Driven Semantic Web, 2004.

[26] Gerti Kappel, Elisabeth Kapsammer, Horst Kargl, Gerhard Kramler, Thomas Reiter, Werner Retschitzegger, Wieland Schwinger, and Manuel Wimmer. Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages. In Proc. of the ACM/IEEE 9th Int'l Conf. on Model Driven Eng. Languages and Sys., pages 528–542, 2006.

[27] Christoph Kiefer, Abraham Bernstein, and Jonas Tappolet. Analyzing Software with iSPARQL. In Proc. the 3rd ESWC Int'l WSh. on Semantic Web Enabled Software Eng., 2007.

[28] Ren´e Witte, Yonggang Zhang, and Juergen Rilling. Empowering Software Maintainers with Semantic Web Technologies. In Proc. of the 4th European Semantic Web Conference, pages 37–52. Springer, 2007.

[29] C´esar Gonz`alez-P`erez and Brian Henderson-Sellers. An Ontology for Software Development Methodologies and Endeavours, volume Ontologies for Software Engineering and Software Technology, pages 123–151. Springer, 2006.

[30] Anupriya Ankolekar, Katia Sycara, James Herbsleb, Robert Kraut, and Chris Welty. Supporting Online Problem-solving Communities with the Semantic Web. In Proc. of the 15th Int'l conf. on WWW, pages 575–584, 2006.

[31] P. Tetlow, J. Z. Pan, D. Oberle, E. Wallace, M. Uschold, and E. Kendall. Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. W3C Working Draft, 2006.

[32] OMG ODM. Ontology Definition Metamodel, 2006. http://www.omg.org/cgibin/doc?ad/06-05-01.pdf.

[33] Hans-J¨org Happel and Stefan Seedorf. Applications of Ontologies in Software Engineering. In Proc. of the Int'l WSh. on Semantic Web Enabled Software Engineering, 2006.