ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/279062467

Future Research Directions of Software Engineering and Knowledge Engineering

Article *in* International Journal of Software Engineering and Knowledge Engineering · March 2015

DOI: 10.1142/S0218194015500035

CITATION
1

READS
10

1 author:

Haiping Xu
University of Massachusetts Dartmouth
57 PUBLICATIONS **522** CITATIONS

SEE PROFILE

All in-text references underlined in blue are linked to publications on ResearchGate, letting you access and read them immediately.

Available from: Haiping Xu
Retrieved on: 24 August 2016

International Journal of Software Engineering and Knowledge Engineering
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

FUTURE RESEARCH DIRECTIONS OF SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING*

HAIPING XU

Computer and Information Science Department, University of Massachusetts Dartmouth
North Dartmouth, MA 02747, USA
hxu@umassd.edu

Received (18 July 2014)
Revised (22 November 2014)
Accepted (22 November 2014)

Software Engineering (SE) and Knowledge Engineering (KE) are closely related disciplines with goals of turning the development process of software systems and knowledge-based systems, respectively, into engineering disciplines. In particular, they together can provide systematic approaches for engineering intelligent software systems more efficiently and cost-effectively. As there is a large overlap between the two disciplines, the interplay is vital for both to be successful. In this paper, we divide the intersection of SE and KE into three subareas, namely Knowledge-Supported Software Engineering (KSSE), Engineering Knowledge as a Software (EKaaS), and Intelligent Software System Engineering (ISSE). For each subarea, we describe the challenges along with the current trends, and predict the future research directions that may have the most potential for success.

Keywords: Software engineering; Knowledge engineering; Knowledge-supported software engineering; Intelligent software systems; Knowledge as a software.

1. Introduction

To enhance the quality of software systems and effective management of software development processes, the discipline of Software Engineering (SE) was first coined in a NATO sponsored international conference in 1968 [1]. The initial goal of software engineering was to apply engineering disciplines to the software development process, in order to deal with the so-called "software crisis" problem. Issues related to the "software crisis" problem include poor software quality, project time and budget being out of control, and difficulties in software maintenance due to a lack of systematic, rigorous and measurable development methodologies. During the past decades, many of the software engineering principles and best practices have been successfully applied to different aspects of a software development process such as requirement analysis, software design, software testing, software maintenance, and software quality assurance [2]. On the other hand, in the early 1980s, much of the research in Artificial Intelligence (AI) focused on the development of Knowledge-Based Systems (KBSs) including expert systems,

* This paper is based on an invited talk of the same title given at the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE 2014), Vancouver, Canada, July 1-3, 2014.

supported by proper knowledge representation formalisms and efficient inference mechanisms. However, the development of large commercial KBSs failed in many cases, which is comparable to the similar situation of "software crisis" in developing traditional software systems in the late 1960s [3]. As a result, the new discipline Knowledge Engineering (KE) emerged with its goal of applying engineering disciplines to the development of scalable and large long-living commercial KBSs. Researchers in KE initially considered the development of KBSs a transfer process that converts existing human knowledge into knowledge bases, but later realized that such a transfer process view could not capture an expert's tacit knowledge for problem-solving capabilities. Thus, the process of building KBSs should be more appropriately viewed as a modeling activity during the knowledge-acquisition phase [3]. Since both KE and SE are essentially modeling processes, they must share many useful principles and methodologies in system modeling and engineering. In the following sections, we divide the intersection of SE and KE into a number of subareas, and then discuss about them in more details.

2. Intersection between Software Engineering and Knowledge Engineering

Figure 1 shows a mapping of the relationship between SE and KE. As shown in the figure, the intersection of SE and KE is identified as SEKE (Software Engineering and Knowledge Engineering), which represents the major topic of this paper. There are many books, journals and proceedings published in the past two decades that are related to SEKE. Among them, the *International Conference on Software Engineering and Knowledge Engineering* (SEKE) and its associated *International Journal of Software Engineering and Knowledge Engineering* (IJSEKE) particularly welcome papers for sharing methods and results between the two disciplines. A central theme of the IJSEKE journal is the interplay between software engineering and knowledge engineering, for example, how knowledge engineering methods can be applied to software engineering, and vice versa [4]. This emphasis makes the scope of IJSEKE different from that of pure SE journals such as *IEEE Transactions on Software Engineering* (TSE) and *ACM Transactions on Software Engineering and Methodology* (TOSEM), and pure KE journals such as *IEEE Transactions on Knowledge and Data Engineering* (TKDE) and Data and Knowledge Engineering Journal (DKE).

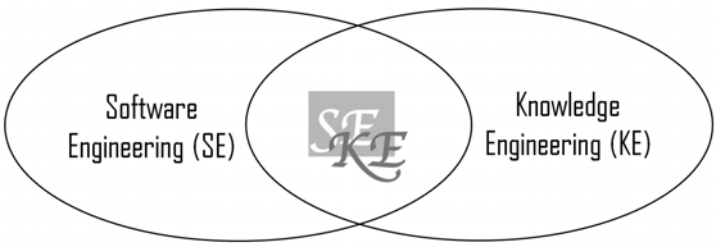

Fig. 1. The intersection of software engineering and knowledge engineering

In order to study the current trends and the future research directions of SEKE, we divide the intersection of SE and KE into three subareas, namely *Knowledge-Supported Software Engineering* (KSSE), *Engineering Knowledge as a Software* (EKaaS), and *Intelligent Software System Engineering* (ISSE). As shown in Fig. 2, the top portion of the intersection, *i.e.*, the subarea KSSE, addresses how knowledge engineering methods can be applied to software engineering; in other words, how to make the software engineering activities more efficient and cost-effective using knowledge-based analysis or knowledge-supported systems. A typical example of research in this area is to analyze software engineering data using big data analysis or data mining approaches. The bottom portion of the intersection in Fig. 2 is called EKaaS, which studies the application of software engineering methods to knowledge engineering. Researchers in this subarea view a knowledge-based system as a traditional software system, and studies how to make the development process of KBSs more efficient and cost-effective by applying software engineering principles and methodologies. An example of such study is to develop large-scale domain knowledge using modularization principle in SE. In the middle portion of the intersection in Fig. 2, the subarea is identified as ISSE, which focuses on engineering intelligent software systems using both SE and KE methods. There are many different types of intelligent software systems nowadays, among which some typical examples include knowledge-based software information systems [5], ambient intelligent systems [6], and slow intelligent systems [7].

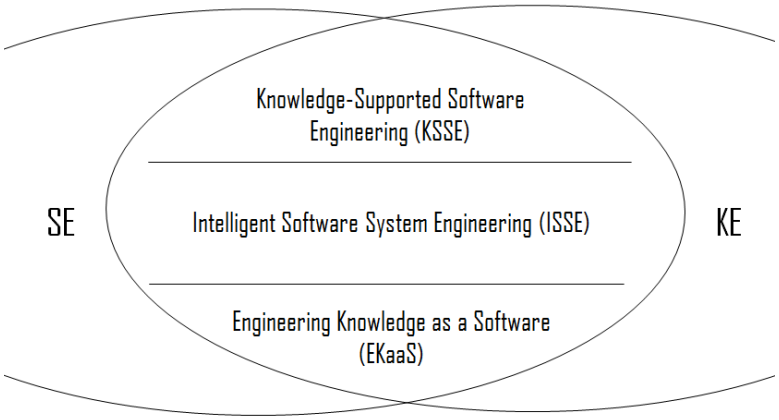

Fig. 2. Subareas in the intersection of software engineering and knowledge engineering

3. Challenges and Future Research Directions

Current prevailing technologies such as mobile cloud computing, big data analysis and ambient intelligence, will have strong impacts on the future research directions of SEKE. In the following sections, we study the current and emerging trends in each subarea of SEKE, and try to predict the possible future research directions in the coming years.

3.1. Intelligent Software System Engineering (ISSE)

The initial goal of AI was to build intelligent machines that may replace human beings; however, researchers later realized that such a goal was not practical and realistic, at least for the time being. Therefore, in the 1990s, agent-based techniques became one of the major themes of AI for the purpose of building intelligent systems that could help human beings solve problems rather than replace them. Although agent-based technologies have been quite successfully used to build intelligent software systems in the past two decades, new computer-based technologies will inevitably bring us new challenges. For example, in a recent prediction by Ann Mack, a former Director of Trendspotting at JWT Worldwide, the smartphone screens will eventually disappear due to the rise of the Internet of Things (IoT) and wearable technology [8]. It is foreseeable that with the wearable devices, such as smartwatches and smartglasses, becoming much popular, more and more intelligent software systems will be shifted from desktops to mobile devices and the cloud. Thus how to efficiently and cost-effectively develop intelligent software systems on mobile devices as well as in the cloud will be a challenging research topic in the next decade. On the other hand, big data analysis and data mining techniques offer us a great opportunity to extract useful knowledge from massive unstructured data, which is critical for predicting future events or user behaviors in constructing practical intelligent software systems. A good example of using such techniques in a smart software system is to achieve predictive security that can track and predict cyber threats, in order to prevent the system from being attacked. This approach can bring obvious advantages over the traditional reactive security, where only defensive actions can be taken when a threat has been identified. There are many other possible future research directions in the subarea ISSE, for example, it would be challenging to study how to effectively develop slow intelligent systems that can gradually learn the way how human solves problems without extensive training [7]. As another example of current trends, Ambient Intelligence (AmI) allows high technology to be seamlessly embedded in our natural surroundings. As a result, users could use such technology in their daily life with effortless interactions [6]. This type of smart systems may present software engineers new critical system requirements, for example, a scalable software architecture that can integrate many different advanced technologies. It is predictable that researchers in this subarea will face many interesting challenges in the next few years.

3.2. Knowledge-Supported Software Engineering (KSSE)

The goal of KSSE is to improve software development process using knowledge engineering methods. Since the success of SE heavily depends on the experience of the software developers, Experience Factory (EF) [9] has become a useful approach in Knowledge-Based Software Engineering (KBSE). Related to this effort, Learning Software Organization (LSO) studies how to improve software process through knowledge discovery, sharing and reuse within software organizations [10]. With the current prevailing technologies like semantic web and ontological engineering, LSO will

continue to be an important research area in the next 10 years. Similarly, Computational Intelligence (CI) and Knowledge Discovery and Data mining (KDD) techniques also played important roles in fields such as software quality assurance, discovery of software defects and project management. To make a software process more efficient and cost-effective, KE methods can be applied in many phases of a software process model. A recent version of Guide to the Software Engineering Body of Knowledge (SWEBOK Guide V3.0) defines 15 Knowledge Areas (KAs) within the field of software engineering. It characterizes the contents of the software engineering discipline, and provides a general framework to apply KE in each KA [11]. In addition, as big data analysis techniques and data mining approaches being increasingly used to acquire useful knowledge from massive unstructured data, effectively mining software engineering data will be an interesting research topic in the coming years [12]. Model checking has been one of the most successful approaches in software engineering during the past decades; however, it is not scalable due to limitations related to the state-explosion problem. Thus, there is a pressing need to apply knowledge-based techniques to make the model checking approach more scalable and practical for verifying software systems of reasonable sizes. As another example, we anticipate in the near future, smart IDEs will emerge to support efficient software development. Such IDEs may automatically acquire knowledge from experienced software developers, and detect major program errors, especially in concurrent programming.

3.3. Engineering Knowledge as a Software (EKaaS)

EKaaS is to use software engineering methods to solve knowledge engineering problems. KE is essentially a modeling process, which shares many common methodologies with SE [3]. For example, the conceptual models in KE are similar to the software architectures in SE; therefore, principles for the development of software architectures may be applied to the construction of conceptual models in KE. Similarly, modularization of domain knowledge follows the same idea of Component-Based Software Engineering (CBSE). It is expected that with well-defined interfaces between different modules of the domain knowledge established by various individuals or organizations, knowledge conflict, inconsistency and uncertainty among different modules may be properly and efficiently resolved. When large-scale domain knowledge being developed, knowledge defined previously must be reused and possibly required to be transformed. This may lead to an interesting research direction that studies how software reuse methodologies could be applied to the reuse of domain knowledge. Finally, formal methods play critical roles in both SE and KE. Some formal techniques that have been successfully applied in SE, such as Petri nets and model checking techniques, may also find their ways in KE to support efficient reasoning and formal verification of domain knowledge.

4. Concluding Remarks

Although software engineering and knowledge engineering are two different disciplines under the subject of computer science, they are closely related and overlapped. As more

and more knowledge being acquired using knowledge engineering methods, many of the software systems can be implemented as intelligent software supported by knowledge bases and efficient reasoning mechanisms. The development of such software systems requires both of the SE and KE disciplines, thus the interplay between SE and KE has become much more important than ever before. In this paper, we divide the intersection of SE and KE into three subareas, discuss about the current trends in each subarea, and predict their possible future research directions. It is our belief that with the new technologies, such as big data analysis and mobile cloud computing, getting more popular, the future research directions of SEKE will be inevitably influenced.

Acknowledgments

The author appreciates the support and guidance from Dr. S. K. Chang, Editor-in-Chief of IJSEKE, and the anonymous reviewers who help improve the paper.

References

[1] P. Naur and B. Randell (Eds.), *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee*, Garmisch, Germany, October 7-11, 1968.

[2] C. L. Simons, I. C. Parmee and P. D. Coward, 35 years on: to what extent has software engineering design achieved its goals?, *IEE Proceedings - Software*, **150**(6) (2003) 337-350.

[3] R. Studer, V. R. Benjamins and D. Fensela, Knowledge engineering: principles and methods, *Data & Knowledge Engineering*, **25**(1–2) (1998) 161–197.

[4] S. K. Chang, Foreword, *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, **1**(1) (1991).

[5] P. Devanbu, P. G. Selfridge, B. W. Ballard and R. J. Brachman, A knowledge-based software information system, in *Proc. of the 12th international conference on Software engineering (ICSE'90)*, Nice, France, March 26-30, 1990, pp. 249-261.

[6] T. Basten, L. Benini, A. Chandrakasan, *et al.*, Scaling into ambient intelligence, in *Proc. of the 6th Design, Automation, and Test in Europe Conference (DATE'03)*, Munich, Germany, March 2003, pp. 76-83.

[7] S. K. Chang, A general framework for slow intelligence systems, *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, **20**(1) (2010) 1-16.

[8] J. Widman, 8 technologies that are on the way out – and one that we'll never be rid of, *Computerworld*, International Data Group Inc., June 20, 2014. Retrieved on June 23, 2014 from http://www.computerworld.com/slideshow/detail/152280

[9] V. R. Basili1, G. Caldiera and H. D. Rombach, Experience factory, *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc., 2002.

[10] A. Birk, T. Dingsøyr, Trends in learning software organizations: current needs and future solutions, *Professional Knowledge Management*, LNCS 3782, Springer Berlin Heidelberg, 2005, pp. 70-75.

[11] P. Bourque and R.E. Fairley (Eds.), *Guide to the Software Engineering Body of Knowledge*, Version 3.0, IEEE Computer Society, 2014.

[12] A. E. Hassan and T. Xie, Software intelligence: the future of mining software engineering data, in *Proc. of the FSE/SDP Workshop on Future of Software Engineering Research (FoSER 2010)*, November 7-11, 2010, Santa Fe, NM, USA, pp. 161-166.