On the Frontiers of Software Science and Software Engineering

Article *in* Frontiers in Computer Science · September 2021

CITATIONS
0

READS
55

1 author:

Yingxu Wang
The University of Calgary
798 PUBLICATIONS **11,823** CITATIONS

SEE PROFILE

frontiers in Computer Science | Software

Y. Wang (2021). On the Frontiers of Software Science and Software Engineering. *Frontiers in Computer Science – Software*. 1(1):1-4.

On the Frontiers of Software Science and Software Engineering

Yingxu Wang, *FIEEE, FBCS, FI2CICC, FAAIA, FWIF, PEng*
Specialty Chief Editor, Frontiers in Computer Science - Software
President, International Institute of Cognitive Informatics and Cognitive Computing (I2CICC)
Dept. of Electrical and Software Engineering
Schulich School of Engineering and Hotchkiss Brain Institute, University of Calgary, Canada
Email: yingxu@ucalgary.ca

Abstract

Advances in software engineering, software science, computational intelligence, and intelligent mathematics have led to the establishment of *Frontiers in Computer Science – Software* (FCSS). FCSS aims to promote transdisciplinary research and engineering applications in software, autonomous systems, and computational intelligence. FCSS covers not only classical empirical software engineering and industrial processes, but also contemporary topics of software science, intelligent programming languages, autonomous code generation, mathematical foundations, and programming knowledge bases. FCSS prompts empirical studies and emerging topics in software engineering including tools, development platforms, industrial processes, management infrastructures, quality assurance schemes, big data systems, and software migrations across languages and platforms.

Keywords: *Software science, software engineering, theories, laws, architectures, requirement analyses, design processes, intelligent mathematics, autonomous systems*

1. Introduction

The latest advances in computer science, software theories, intelligence science, intelligent mathematics, and autonomous systems have triggered the emergence of software science as a transdisciplinary field overarching software engineering, programming theories, formal methods, and computational intelligence [1-8].

Software science studies the formal properties and mathematical models of software, general methodologies for rigorous and efficient software development, and coherent theories underpinning software behaviors and software engineering practices [6]. The discipline of software science encompasses theories and methodologies, denotational mathematics, system software, fundamental algorithms, organizational theories, cognitive complexity of software, and intelligent behavior generation theories. Recent developments in software science have paved the way to enable novel technologies for AI programming.

Software engineering is a discipline underpinned by software science that adopts engineering approaches to develop large-scale software towards high productivity, low cost, trustworthy quality, and controllable development schedule [7]. Software engineering is one of the most complicated branches of engineering fields because its object is highly abstract and intangible. It encompasses system modeling, architecting, development methodologies, programming technologies, and development platforms. It also covers heuristic principles, tools/environments, best practices, case studies, experiments, trials, and performance benchmarking.

This editorial addresses the objectives, mission, challenges, and latest development of *Frontiers in Computer Science – Software* (FCSS). In the framework of FCSS, software science focuses on foundations and theoretical software engineering, whilst software engineering covers heuristic principles, tools, development environments, best practices, and programming technologies. A set of key challenges and opportunities of software science and engineering has been recognized in Section 2. The emerging fields of software science and novel technologies of software engineering are elaborated in Sections 3 and 4, respectively.

2. Challenges and Opportunities in Software Science and Engineering

Despite the fast growth of software engineering over the past 60 years, software development has still remained as a manual work in the software industry [3, 4, 6]. This phenomenon indicates a fundamental challenge to both theories and technologies of *Software Science and Engineering* (SSE), which are substantially constrained by the following observations:

a) The fundamental theories towards software science are immature;
b) The traditional perceptions on the essences of software or programs are questionable;
c) The classic human-based programming technologies are anti-productive;
d) Essential intelligent mathematics for modeling and manipulating human and software behaviors are missing;
e) The current programming languages constrained by deterministic condition-driven mechanisms have seriously limited the programable power of software for autonomous intelligence generation.

A set of *Key Fundamental Challenges* (KFC) to SSE are identified in basic research findings [1-14, 21-24] in this field represented by the following grand queries:

1) What are the necessary and sufficient conditions for enabling run-time software intelligence in SSE?
2) Why had AI and autonomous systems been developed by data-driven neural networks rather than program-driven software engineering?
3) Does that of KFC2 indicate a theoretical or technical challenge to SSE?
4) How mature are our computing platforms and programming languages for enabling autonomous software generation?
5) Is *Stored-Program-Controlled* (SPC) computers [1,2], i.e., von Neumann machines (VNM) [2], adequate enough for intelligent software design? If not, what kind of computers will be needed for the next generation of intelligent software engineering?
6) Are our programming languages sufficiently expressive for designing intelligent software? What would happen to a software system if the deterministic if-then-else structures were indeterminable at designed-time or exhausted at run-time?
7) Are our mathematical means ready for rigorously expressing software system requirements? Is our inference power adequate for expressing real-time inexhaustive, indeterministic, and uncertain behaviors in SSE?
8) How may an intelligent software system be trusted and verified when its state space is infinitive in de facto, such as those of self-driving vehicles and mission-critical robots?
9) How may a nondeterministic intelligent programming language be created in order to enable run-time intelligent behavior for handling indeterministic events and autonomous decision-making requirements?
10) How will SSE enable the next generation of intelligent system software and autonomous code generation systems?

The KFCs provide a set of theoretical and empirical challenges to and opportunities for SSE as well as the software industry. Any breakthrough on KFCs will lead to a wide range of breakthroughs and applications in the software and computational intelligence industries.

3. The Frontier of Software Science

It is recognized that a rigorous theoretical framework of software science is yet to be sought despite the plenty repository of empirical knowledge about software development. As Edsger Dijkstra stated that “Software engineering is programming when you can't [3].” Therefore, software science is what you can’t by empirical software engineering [6]. The pinnacle of software science is the theories of formal methods [1,4,5,8] and platforms of system software including operating systems, compilers, database management systems, and Internet-based distributed systems [2,3,6]. C.A.R. Hoare has revealed that software is a *mathematical entity* [4] that may be formally denoted by a behavioral process known as the *unified process theory* in formal methods towards software engineering [5]. A set of 30+ laws of programming has been created by Hoare and his colleagues [5]. Inspired by his genius vision, the author has developed the *Real-Time Process Algebra* (RTPA) [16], where who was a visiting professor to Prof. Hoare at University of Oxford in 1995. RTPA extends Hoare’s sequential process theory to a system of intelligent mathematics for formally denotating system and human cognitive, inference, and behavioral processes as a general theory towards *software science* [6]. Based on RTPA, a comprehensive set of 95 mathematical laws and associated theorems on software structures, behaviors, and processes has been formally established [8].

Software science is a discipline that studies the formal properties and mathematical models of software, general methodologies for rigorous and efficient software development, and coherent theories and laws underpinning software behaviors, as well as software engineering practices. The architecture of software science encompasses theories and methodologies, intelligent mathematics, intelligent system software, and software engineering processes [7]. It is

noteworthy that the focuses of software science are on the fundamental platforms of *Intelligent System Software* (ISS) including those of intelligent operating systems, intelligent requirement specifiers, intelligent compilers, and intelligent programming knowledge bases. The development of ISS will be based on an indispensable foundation known as intelligent mathematics that enables software engineering to be matured towards software science based on rigorous denotational mathematics beyond inexpressive programming languages.

Intelligent mathematics (IM) [14, 15] is centric in software science as a category of denotational mathematical structures. IM deals with complex mathematical entities beyond pure numbers such as abstract objects, complex relations, behavioral information, concepts, knowledge, processes, programmable intelligence, and systems. IM is an indispensable foundation for dealing with the abstract and complex entities of software structures and behaviors. A set of IM have been created in the last decades embodied by *system algebra* [17], *concept algebra* [18], RTPA [16], *semantic algebra* [19], *inference algebra* [20], and etc. IM provides a coherent set of contemporary mathematical means and explicit expressive power for manipulating both complex mathematical objects and long-chains of serial and embedded mathematical operations emerged in applied disciplines. Therefore, the maturity of IM indicates the maturity of software science for rigorous processing system architectures and behaviors with abstract concepts, complex relations, and dynamic processes [14]. In software science, a General Mathematical Model of Software (GMMS) [6] is discovered based on RTPA that reveals any software system is a derived instance of GMMS for rigorously manipulating the *Structure Models* (SMs) and *Process Models* (PMs) of arbitrary software as behavioral interactions in the *universe of discourse of software* Ω determined by the Cartesian product $\Omega = PM \times SM$ [6].

4. The Frontier of Software Engineering

The latest advances of theories and methodologies in software science, computational intelligence, and IM have paved a way to enable machines to generate programs in software engineering [7]. The challenges to intelligent program generation were constrained by the lack of autonomous programming theories and the difficulties stemmed from the complexity of software. The missing of machine-understandable semantical rules of software behaviors has prevented software engineering from advancing to intelligent program generation. Pilot studies indicate that a fusion of the analytic (mathematical rule-based) and gray-box (machine-learning-based) methodologies will be necessarily adopted towards autonomous program generation [23, 24].

An ultimate strategy for the next generation of software engineering is AI program (AIP) [24], which has been enabled by a comprehensive set of software laws for software architectures, structures, and behaviors. In the AIP approach to software engineering, IM services as rigorous mathematical means for conveying human programming knowledge and skills to machines [6]. IM for AIP is underpinned by: a) System algebra for software architectural design; b) Concept algebra and semantic algebra for rigorous requirement modeling; and c) RTPA for formally manipulating software structures and behaviors. Not only the architectures of computational intelligent systems, but also their dynamic behaviors can be rigorously and systematically manipulated by IM. Several large-scale projects designed based IM have demonstrated that it is a set of powerful mathematical means for dealing with concepts, knowledge, behavioral processes, and human/machine intelligence in real-world software engineering.

The relationship between software science and software engineering is explained by analogizing those of pure and applied physics. Without theoretical physics there would be no maturity of applied physics; so is software science for software engineering. It is recognized that the phenomena where almost all the fundamental problems that could not be solved in the past 60 years in software engineering has been stemmed from the lack of coherent theories in the domain of software science toward autonomous software generation.

5. Conclusion

The emerging field of software science and engineering (SSE) investigates into the theoretical foundations and intelligent mathematics for autonomous software generation. This editorial has explored the latest basic research in software science. Applications of software science and intelligent mathematics in software engineering and autonomous code generation have been presented. A set of key challenges and opportunities for SSE has been formally reviewed and analyzed towards intelligent software engineering.

Acknowledgement

Many people have contributed to the establishment of FCSS. The Specialty Chief Editor (SCE) of FCSS would like to thank the EIC of FCS, Prof. Kaleem Siddiqi, associate editors, regular reviewers, and authors. The SCE of FCSS would like to acknowledge the professional initiative and support of Dr. Thomas Croft, Dr. Enrique Morillas, and Dr. Bennett Colgan at the editorial office of Frontiers.

References

[1] A.M. Turing (1950). Computing machinery and intelligence. Mind (59):433–460.
[2] J. von Neumann. (1946). The Principles of Large-Scale Computing Machines. *Annals of History of Computers*, 3(3), 263-273
[3] E.W. Dijkstra (1976), *A Discipline of Programming*. Prentice Hall, Englewood Cliffs, NJ.
[4] C.A.R. Hoare (1978). Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677.
[5] C.A.R. Hoare, I.J. Hayes, J. He, etc. (1987). Laws of Programming. *Communications of the ACM*. 30(8):672–686.
[6] Y. Wang (2014). Software Science: On General Mathematical Models and Formal Properties of Software, *Journal of Advanced Mathematics and Applications*, 3(2):130-147.
[7] Y. Wang (2007). *Software Engineering Foundations: A Software Science Perspective*. CRC/Auerbach Publications, NY, USA, 1,580pp.
[8] Y. Wang (2008). Mathematical Laws of Software. *Transactions of Computational Science*, Springer, 2:46-83.
[9] Y. Wang, F. Karray, O. Kaynak, et al. (2021). Perspectives on the Philosophical, Cognitive and Mathematical Foundations of Symbiotic Autonomous Systems. *Philosophical Transactions of Royal Society* (A). Oxford, UK, 379(2207):1-20.
[10] Y. Wang (2002). Cognitive Models of the Brain. *Proceedings First IEEE International Conference on Cognitive Informatics* (ICCI'02), IEEE CS Press, Aug., pp. 259-269.
[11] Y. Wang (2007). the Cognitive Processes of Formal Inferences. Int'l Journal of Cognitive Informatics & Natural Intelligence. 1(4):75-86.
[12] Y. Wang (2009). Formal Description of the Cognitive Process of Memorization. *Transactions on Computational Science*, Springer, 5:81-98.
[13] Y. Wang, D. Liu, and G. Ruhe (2004). Formal Description of the Cognitive Process of Decision Making. *Proceedings of the 3rd IEEE International Conference on Cognitive Informatics*, IEEE CS, Press, pp.124-130.
[14] Y. Wang (2020), Keynote: Intelligent Mathematics: A Basic Research on Foundations of Autonomous Systems, General AI, Machine Learning, and Intelligence Science, *IEEE 19th Int'l Conf. on Cognitive Informatics and Cognitive Computing* (ICCI*CC'20), Tsinghua Univ., Beijing, China, Sept., p.5.
[15] Y. Wang (2012), In Search of Denotational Mathematics: Novel Mathematical Means for Contemporary Intelligence, Brain, and Knowledge Sciences, Journal of Advanced Mathematics and Applications, 1(1), 4-25.
[16] Y. Wang (2008). RTPA: A Denotational Mathematics for Manipulating Intelligent and Computational Behaviors, *Int'l Journal of Cognitive Informatics & Natural Intelligence*, 2(2):44-62.
[17] Y. Wang (2008). On System Algebra: A Denotational Mathematical Structure for Abstract Systems, Modeling, *Int'l Journal of Cognitive Informatics & Natural Intelligence,* 2(2):20-43.
[18] Y. Wang (2006). On Concept Algebra and Knowledge Representation. 5th IEEE Int'l Conference on Cognitive Informatics (ICCI'06). vol. I., 320-331.
[19] Y. Wang (2013). On Semantic Algebra: A Denotational Mathematics for Cognitive Linguistics, Machine Learning, and Cognitive Computing, *Journal of Advanced Mathematics and Applications*, 2(2):145-161.
[20] Y. Wang (2012). Inference Algebra (IA): A Denotational Mathematics for Cognitive Computing and Machine Reasoning (II), Int'l Journal of Cognitive Informatics & Natural Intelligence, 6(1):21-47.
[21] Y. Wang (2008). On the Big-R Notation for Describing Interactive and Recursive Behaviors, *International Journal of Cognitive Informatics and Natural Intelligence*. 2(1):17-28.
[22] Y. Wang (2012). On Visual Semantic Algebra (VSA): A Denotational Mathematical Structure for Modeling and Manipulating Visual Objects and Patterns. *Software and Intelligent Sciences: New Transdisciplinary Findings*, pp. 68-81.
[23] Y. Wang, Y. (2004), On Cognitive Informatics Foundations of Software Engineering, *3rd IEEE Int'l Conference on Cognitive Informatics* (ICCI'04), IEEE CS Press, Canada, pp. 22-31.
[24] J.Y. Xu and Y. Wang (2020), Formal Requirement Analysis and Specification Modeling for AI Software Generation, *IEEE 19th Int'l Conf. on Cognitive Informatics and Cognitive Computing* (ICCI*CC'20), Tsinghua Univ., Beijing, China, Sept., pp. 187-194.