Responsible Software Engineering

Ina Schieferdecker

Abstract Software trustworthiness today is more about acceptance than technical quality; software and its features must be comprehensible and explainable. Since software becomes more and more a public good, software quality becomes a critical concern for human society. And insofar artificial intelligence (AI) has become part of our daily lives—naturally we use language assistants or automatic translation programs—software quality is evolving and has to take into account usability, transparency as well as safety and security. Indeed, a majority worldwide rejects currently the use of AI in schools, in court or in the army because it is afraid of data misuse or heteronomy. Insofar, software and its applications can succeed only if people trust them. The initiatives towards "responsible software engineering" address these concerns. This publication is about raising awareness for responsible software engineering.

Keywords Software testing · Software quality · Software engineering

1 Introduction

Regardless of whether it is an autonomous vehicle, an artificial intelligence or a mobile robot—software is always steering these systems and is a key component in digital transformation. The same applies to critical infrastructures that shape the lives of millions of people: utilities for electricity, water and gas, but also large parts of the transport infrastructure are already based on information and communication technologies (ICT). Scenarios for smart homes, smart manufacturing and smart cities are even further extending the influence of software on our everyday lives [1]: "The software industry directly contributed €304 billion to the EU economy in 2016, representing 2% of total EU value-added GDP—up 22.4% from €249 billion in 2014. The sector employed 3.6 million people and paid €162.1 billion in wages.

I. Schieferdecker
Federal Ministry of Education and Research, Berlin, Germany

Software companies were responsible for an average of 1.8% of total jobs in the seven EU countries in this study."

By that, software leaves the pure technical system management and control and serves today increasingly for decision support and decision control in socially critical contexts. The questions that arise are which processes are necessary, how to make the software comprehensible and who bears the responsibility. And how can companies ensure the reliability, quality and security of software in these increasingly complex environments?

Even further, not only in the use, even in the production and distribution of software, the expectations of customers are exhaustive. Hence, software development is to go faster and faster, the product at the same time getting better and better. And this expectation is not necessarily paradoxical. In many cases, DevOps' holistic approach can actually increase both the speed of delivery and the quality. Where the word "deliver" in times of cloud computing does not really characterize the process anymore. After all, the customer no longer buys a product, but instead access to a service. Even—and especially—such a service will only be accepted if users trust it.

Important elements in establishing trust are so-called testbeds for field experiments and experimental environments for co-innovation. Traceability and transparency should be part of software development. Because the complexity of software-based systems continues to grow, and because data retention and use is often difficult to understand, trust in software is still often fragile. Hence, principles of anti-fragility in software have to be added, which add fault resilience and robustness at run time [2]. Also, the traceability of goals, features and responsibilities need to become part of any software engineering and documentation. Why this is so important is explained in this quote from computer scientist Joseph Weizenbaum: "A computer will do what you tell it to do, but that may be much different from what you had in mind". And although methods and tools for high-quality, reliable and secure software development are available in large numbers, they are to be extended to address runtime faults as well [3].

The chapter starts with a general consideration of software and current pressing issues. This is followed by a review of the current understanding of ethical principles in software engineering and draws conclusions towards responsible software engineering. A summary and outlook complete the chapter.

2 Software and Recent Software Quality Requirements

Software is basically a set of instructions that tells a computer, embedded control systems or a (micro-)processor what to do and how to do it. Software is not just a programming code on firmware, operating system, middleware or application level. It also consists of data that represent the content managed by the programs as well as data that train or steer the programs [4]. In addition, it encompasses meta-data that represent information and documentation of the software [5]. According to ISO [6], software is a "fundamental term for all or part of the programs, procedures, rules, and associated documentation of an information processing system. . . . (It) is an intellectual creation that is independent of the medium on which it is recorded."

Software exists in many types and variants and there is no widely adopted software taxonomy. Rather, there exist surveys for specific fields of software applications or software development tools such as in software-defined networks [7], for user interfaces [8] or for software documentation [9].

Although it is apparently hard to grasp characteristics of software in general, there is a long-lasting and still growing understanding of how software quality is constituted. The ISO 25010 provides an updated set of software quality requirements [10] compared to ISO 9126 or other previously established software quality models, like the one by Boehm et al. [11], FURPS [12], by IEEE [13], Dromey [14] or QMOOD [15].

Still, since software technologies evolve, the understanding of software quality needs to evolve as well. For example, software usability addresses the ease of use of a software [16]. For specified consumers, it seeks to improve their effectiveness, efficiency and satisfaction in achieving their objectives by the given context of use and the usage scenarios for a software.

Another example is the growing importance of data (as software in itself and as a software artefact in use) in big data, Internet of Things or artificial intelligence applications [17]. Data quality assessment [18] aims at deriving objective data quality metrics that resemble also subjective perceptions of data.

Let us also refer to the growing use of software in emulating reality in virtual and augmented reality applications in gaming, for education or for training [19, 20]. By that, multimedia (streams) in 2D, 3D and eventually 4D contexts in presentation, but also in interactive modes, require more elaborated media, interaction and collaboration attributes in software quality.

As a final example let us consider the growing need for transparency of software so that users receive a solid understanding about what a software provides and what it does not provide. The more software permeates into every field of our society, transparency, traceability and explainability become the central quality criteria that also demand attention from software developers [21].

3 Software Criticality and the Need for Responsible Software Engineering

To the extent that society is increasingly dependent on autonomous, intelligent and critical software-based systems in energy supply, mobility services and production, but also in media discourses or democratic processes, new strategies must be found to ensure not only their well-understood quality characteristics such as safety, efficiency, reliability and security, but also their associated socio-technical and socio-political implications and all additional requirements in the context of human-machine interaction and collaboration [22].

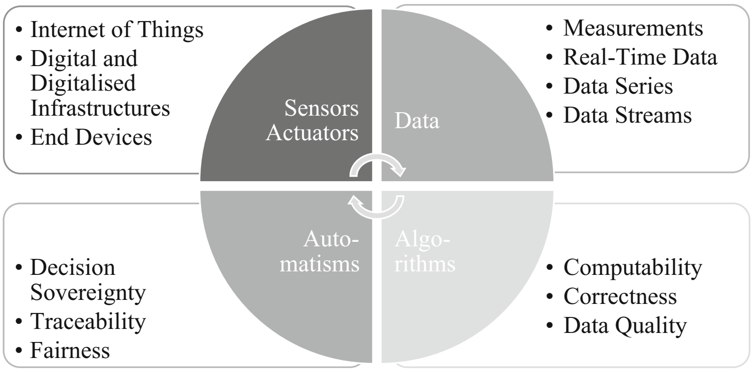

Fig. 1 Elements of software-based systems [23]. Sensors are part of the Internet of Things and generate different kinds of data such as measurements, series of measurements or data streams. Algorithms use these data in their computations or as training data. The algorithms are constrained by complexity, computability and performance limits and possibly by the (in-)correctness of the implemented computation logic and by the (un-)biased (training) data. As a result, software-based systems offer automatisms for which it is essential to agree (and assure) decision sovereignty, traceability and fairness. Any decision in respect to the environment can finally be fed via software (into the cyberspace) and via actuators (into the environment)

Such software-based systems use functionalities as being defined in (meta-) algorithms and steered by data (see Fig. 1).

These software-based systems are also called algorithm-based or algorithmic systems [23]. They are being used for decision-making support or decision-making a in socio-critical context (e.g. in elections), business-critical context (e.g. in online trading) and relevant to self-determination for individuals, organizations and nations. This raises the discussion about the necessary guidelines for the design, development and operation of these software-based systems, which must be understood in the interplay of technological, social and economic processes. They are and become increasingly critical for the whole human society and developed into a public good [23].

In fact, most of the values designed and encoded into these systems stem from the software engineered by the business owners, product owners, software designer and/or software engineers [24]. Software engineering is constituted mainly by (1) defining and constraining the software (requirements engineering and software specification), (2) designing and implementing the software (coding), (3) verifying and validating the software (simulation, model checking, testing, etc.) and (4) operating, maintaining and evolving the software. Software engineering does not need to follow a line of software engineering methods [25], but rather a line of value concerns [23]: Responsible Software Engineering should be constituted by:

1. *Sustainability by Design* by people in power: A critical examination of these value inscriptions should serve as the basis for conscious, reflected valuations, also in order to realize values from the sustainability context. In addition to

the promotion of privacy, safety and security, and quality through appropriate software engineering, sustainability should be anchored in software engineering, for example (1) the ecological sensitivity for energy and resource efficiency of software and (2) the value-sensitivity in data collections, algorithms and heuristics.
2. *Techno-Social Responsibility* by the software community: Not only corporate social responsibility [26] should be addressed by the digital community, but also a techno-social responsibility in the meaning of (1) understanding how digital business models could as well as should not affect society and (2) shaping the digital business models, solutions and infrastructures according to the agreed societal principles.
3. *Responsible Technology Development* by the society: Responsible software engineering should be strategically promoted and supported by appropriate research funding, also known as Responsible Research and Innovation [26] in the meaning of research and innovation (1) based on societal goals, which should also (2) explicitly anchor and demand the UN sustainable development goals [27].
4. *State-of-the-Art Software Engineering* within every software project: It is in the responsibility of the people in power and in action to make use of those software engineering methods and tools that fit the purpose and that fit the level of software criticality. This is not only a matter of tort liability but also of societal responsibilities in light of safety-, security-, environment-, or business-critical software-based systems.
5. Last but not least, such responsible software engineering (see Fig. 2) could be promoted by a *Weizenbaumian Oath* [28] to reflect the professional ethics for sustainable design, development, operation and maintenance, and use of software and of software-based systems. Joseph Weizenbaum (1923–2008) was a computer science pioneer, who critically examined computer technologies and the interactions of humans and machines. He called for a responsible use of technology. Through the Weizenbaumian Oath, all the tech communities could commit to general principles that guide the development and application use of software and of software-based systems. These principles should also become an integral part of the education and training of experts and may constitute a new module in education schemes in software engineering including ISTQB [29].

4 Ethical Principles in Responsible Software Engineering

In responsible software engineering, in addition to software quality matters, the focus is on the comprehensibility, explainability and fairness of software-based systems, and on the ultimate people's decision sovereignty in critical socio-technical contexts. Professional organizations such as the Association for Computing Machinery (ACM) or the German Association for Informatics (GI) already give guidance

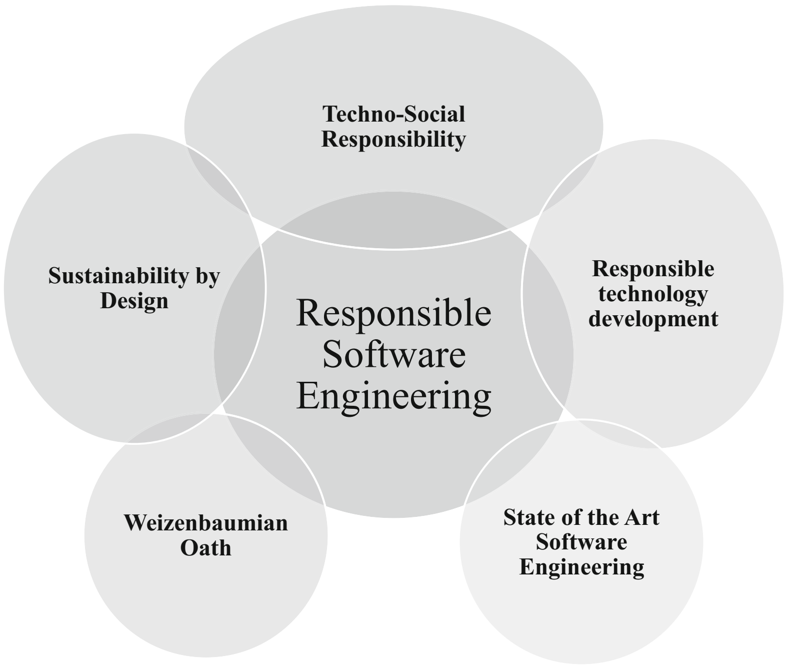
Techno-Social Responsibility
Sustainability by Design
Responsible Software Engineering
Responsible technology development
Weizenbaumian Oath
State of the Art Software Engineering

Fig. 2 Constituents of responsible software engineering

to the tech communities through recently updated ethical guidelines [30, 31]. These and similar initiatives provide a solid basis for the extension of professional ethics towards responsible software engineering.

In view of AI, automation and criticality of software-based systems, the initiatives by the High-Level Expert Group on Artificial Intelligence by the European Commission, by the AI4People and by iRights.Lab explained below provide rule sets for coping with software-based systems and have an understanding of dynamically updating these rule sets in view of ongoing socio-technical and socio-political discourses as well as along rapid technical advancements. Its national, European and international operationalization is an open field for implementation and regulation. In addition, all three represent an urgent need for action because the initiatives remain ineffective if they do not lead to the best possible implementations, which prevent side-effects and unintended risks, in a timely manner [23].

The High-Level Expert Group on Artificial Intelligence by the European Commission is working on Ethics Guidelines for Trustworthy AI [32] and has the following normative foundations:

1. "Develop, deploy and use AI systems in a way that adheres to the ethical principles of: respect for human autonomy, prevention of harm, fairness and explicability. Acknowledge and address the potential tensions between these principles.
2. Pay particular attention to situations involving more vulnerable groups such as children, persons with disabilities and others that have historically been disadvantaged or are at

risk of exclusion, and to situations which are characterized by asymmetries of power or information, such as between employers and workers, or between businesses and consumers.

3. Acknowledge that, while bringing substantial benefits to individuals and society, AI systems also pose certain risks and may have a negative impact, including impacts which may be difficult to anticipate, identify or measure (e.g. on democracy, the rule of law and distributive justice, or on the human mind itself.) Adopt adequate measures to mitigate these risks when appropriate, and proportionately to the magnitude of the risk."

Another initiative is AI4People: It is a multi-stakeholder forum that "brings together all stakeholders interested in shaping the societal impact of AI—including the European Commission, the European Parliament, civil society organizations, industry and the media" [33]. The result is a living document with the following preamble: "We believe that, in order to create a Good AI Society, the ethical ... should be embedded in the default practices of AI. In particular, AI should be designed and developed in ways that decrease inequality and further social empowerment, with respect for human autonomy, and increase benefits that are shared by all, equitably. It is especially important that AI be explicable, as explicability is a critical tool to build public trust in, and understanding of, the technology."

The so-called Algo.Rules [34] define a new approach on how to promote software trust systematically. It was developed by the think tank iRights.Lab together with several experts in the field. New rules define how an algorithm must be designed in order to be able to be evaluated with moral authority: above all, transparent, comprehensible in its effects and controllable:

1. "Strengthen competency: The function and potential effects of an algorithmic system must be understood.
2. Define responsibilities: A natural or legal person must always be held responsible for the effects involved with the use of an algorithmic system.
3. Document goals and anticipated impact: The objectives and expected impact of the use of an algorithmic system must be documented and assessed prior to implementation.
4. Guarantee security: The security of an algorithmic system must be tested before and during its implementation.
5. Provide labelling: The use of an algorithmic system must be identified as such.
6. Ensure intelligibility: The decision-making processes within an algorithmic system must always be comprehensible.
7. Safeguard manageability: An algorithmic system must be manageable throughout the lifetime of its use.
8. Monitor impact: The effects of an algorithmic system must be reviewed on a regular basis.
9. Establish complaint mechanisms: If an algorithmic system results in a questionable decision or a decision that affects an individual's rights, it must be possible to request an explanation and file a complaint."

5 Outlook

Software engineering has changed dramatically since 1968, when it was coined for the first time as an engineering discipline [35]. According to [25]), software engineering has been in the Structured Methods Era 1960–1980 and in the Object Methods Era 1980–2000 and is currently in the Agile Methods Era. These eras not only brought "method wars" and "zig-zag-paths" to software engineering, but also put the focus on technical aspects, software features and methodological approaches rather than putting it on the societal impact of software. In fact, along recent digital transformation discourses, not only the central role of software became apparent to the public, but also the need to find a new framing for software engineering. This framing is coined to be "*responsible software engineering*" in this chapter. It is constituted by five central elements, which are sustainability by design performed by people in power, techno-social responsibility by the software communities, responsible technology development by the society, state-of-the-art software engineering within every software project and the Weizenbaumian oath for all experts.

Responsible software engineering is anticipated by several initiatives that arose from discussions around professional ethics in the software communities specifically as well as by addressing grand challenges in research and innovation in general. It will take time till wide-spread acceptance and deployment, but we need to take actions now and develop approaches and programs which are taught at universities and in industry. Along digital transformation, software and its engineering became public goods and have to be addressed and coped with appropriately. It is not any longer a niche concern, but it is in the interest of us all to design and develop the software also on the basis of a public discourse. In this view, software quality is to be extended along societal impact, transparency, fairness and trustworthiness, which will require not only new or extended methods and tools, but also updated processes and regulations.

Acknowledgments This work has been partially funded by the Federal Ministry of Education and Research of Germany (BMBF) under grant no. 16DII111 ("Deutsches Internet-Institut", Weizenbaum-Institute for the Networked Society) as well as by the German Federal Ministry of Education and Research and the Federal Ministry for the Environment, Nature Conservation and Nuclear Safety under grant number 01RIO708A4 ("German Advisory Council on Global Change", WBGU).

The author thanks the numerous discussions with Stefan Ullrich, Jacob Kröger, Andrea Hamm, Hans-Christian Gräfe, Diana Serbanescu, Gunay Kazimzade and Martin Schüssler all from Weizenbaum-Institute as well as with Reinhard Messerschmidt, Nora Wegener, Marcel J. Dorsch, Dirk Messner and Sabine Schlacke at WBGU.

Last but not least, the author thanks the iSQI team for years of successful and pleasant cooperation to make software quality more present and to offer numerous software quality training schemes that improve the knowledge and expertise in the field. Congrats on its 15th birthday, wishing iSQI at least another 15 successful years of extending the body of knowledge in software quality.

References

1. Unit, E.I.: The growing €1 trillion economic impact of software. (2018)
2. Russo, D., Ciancarini, P.: A proposal for an antifragile software manifesto. Procedia Comput. Sci. **83**, 982–987 (2016)
3. Russo, D., Ciancarini, P.: Towards antifragile software architectures. Procedia Comput. Sci. **109**, 929–934 (2017)
4. Osterweil, L.J.: What is software? The role of empirical methods in answering the question. In: Münch, J., Schmid, K. (eds.) Perspectives on the Future of Software Engineering, pp. 237–254. Springer, Berlin (2013)
5. Parnas, D.L., Madey, J.: Functional documents for computer systems. Sci. Comput. Program. **25**(1), 41–61 (1995)
6. ISO/IEC: Information technology — Vocabulary. 2382. (2015)
7. Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H.: A survey on software-defined networking. IEEE Commun. Surveys Tuts. **17**(1), 27–51 (2014)
8. Myers, B.A., Rosson, M.B.: Survey on user interface programming. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems (1992)
9. Forward, A., & Lethbridge, T.C.: The relevance of software documentation, tools and technologies: a survey. Paper presented at the Proceedings of the 2002 ACM symposium on document engineering (2002)
10. Gordieiev, O., Kharchenko, V., Fominykh, N., Sklyar, V.: Evolution of software quality models in context of the standard ISO 25010. Paper presented at the proceedings of the ninth international conference on dependability and complex systems DepCoS-RELCOMEX, Brunów, Poland, 30 June–4 July 2014 (2014)
11. Boehm, B.W., Brown, J.R., Lipow, M.: Quantitative evaluation of software quality. Paper presented at the Proceedings of the 2nd international conference on Software engineering, San Francisco, California, USA (1976)
12. Grady, R.B., Caswell, D.L.: Software Metrics: Establishing a Company-wide Program. Prentice-Hall, Englewood Cliffs (1987)
13. IEEE: Standard for Software Maintenance. (Std 1219). (1993)
14. Dromey, R.G.: A model for software product quality. IEEE Trans. Softw. Eng. **21**(2), 146–162 (1995)
15. Hyatt, L.E., Rosenberg, L.H.: A software quality model and metrics for identifying project risks and assessing software quality. Paper presented at the product assurance symposium and software product assurance workshop (1996)
16. Seffah, A., Metzker, E.: The obstacles and myths of usability and software engineering. Commun. ACM. **47**(12), 71–76 (2004)
17. Wang, R.Y., Strong, D.M.: Beyond accuracy: what data quality means to data consumers. J. Manag. Inf. Syst. **12**(4), 5–33 (1996)
18. Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. Commun. ACM. **45**(4), 211–218 (2002)
19. Di Gironimo, G., Lanzotti, A., Vanacore, A.: Concept design for quality in virtual environment. Comput. Graph. **30**(6), 1011–1019 (2006)
20. Klinker, G., Stricker, D., Reiners, D.: Augmented reality: a balance act between high quality and real-time constraints. In: Ohta, Y., Tamura, H. (eds.) Mixed Reality: Merging Real and Virtual Worlds, pp. 325–346. Springer, Berlin (1999)
21. Serrano, M., do Prado Leite, J.C.S.: Capturing transparency-related requirements patterns through argumentation. Paper presented at the 2011 first international workshop on requirements patterns (2011)
22. Schieferdecker, I., Messner, D.: The digitalised sustainability society. Germany and the World 2030. (2018)
23. WBGU: Our common digital future. German Advisory Council on Global Change, Berlin (2019)

24. Brey, P.: Values in technology and disclosive computer ethic. In: Floridi, L. (ed.) The Cambridge Handbook of Information and Computer Ethics, pp. 41–58. Cambridge University Press, Cambridge, MA (2010)
25. Jacobson, I., Stimson, R.: Escaping method prison – On the road to real software engineering. In: Gruhn, V., Striemer, R. (eds.) The Essence of Software Engineering, pp. 37–58. Springer, Cham (2018)
26. Porter, M.E., Kramer, M.R.: The link between competitive advantage and corporate social responsibility. Harv. Bus. Rev. **84**(12), 78–92 (2006)
27. Biermann, F., Kanie, N., Kim, R.E.: Global governance by goal-setting: the novel approach of the UN Sustainable Development Goals. Curr. Opin. Environ. Sustain. **26**, 26–31 (2017)
28. Weizenbaum, J.: On the impact of the computer on society: how does one insult a machine? In: Weckert, J. (ed.) Computer Ethics, pp. 25–30. Routledge, London (2017)
29. Strazdiņa, L., Arnicane, V., Arnicans, G., Bičevskis, J., Borzovs, J., Kuļešovs, I.: What software test approaches, methods, and techniques are actually used in software industry? (2018)
30. ACM: ACM code of ethics and professional conduct. Association for Computing Machinery's Committee on Professional Ethics (2018)
31. GI: Unsere ethischen Leitlinien, p. 12. Gesellschaft für Informatik, Bonn (2018)
32. EC: Ethics Guidelines for Trustworthy AI, p. 41. European Commission High-Level Expert Group On Artificial Intelligence, Brüssel (2019)
33. Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Rossi, F., et al.: AI4People—An ethical framework for a good ai society: opportunities, risks, principles, and recommendations. Mind. Mach. **28**(4), 689–707 (2018)
34. iRights.Lab: Algo.Rules: Regeln für die Gestaltung algorithmischer Systeme. Retrieved from Gütersloh, Berlin. https://www.bertelsmann-stiftung.de/fileadmin/files/BSt/Publikationen/GrauePublikationen/Algo.Rules_DE.pdf (2019)
35. Naur, P., Randell, B.: Software engineering-report on a conference sponsored by the NATO Science Committee Garimisch, Germany. https://carld.github.io/2017/07/30/nato-software-engineering-1968.html (1968)