With Great Power Comes Great Responsibility: The Role of Software Engineers

Citation for the original published paper (version of record):

Betz, S., Penzenstadler, B. (2025). With Great Power Comes Great Responsibility: The Role of Software Engineers. ACM Transactions on Software Engineering and Methodology, 34(5). http://dx.doi.org/10.1145/3715112

(article starts on next page)

With Great Power Comes Great Responsibility: The Role of Software Engineers

STEFANIE BETZ, Department of Computer Science, Furtwangen University, Furtwangen, Germany and Lappeenranta-Lahti University of Technology, Lappeenranta, Finland
BIRGIT PENZENSTADLER, Chalmers University of Technology, Göteborg, Sweden, University of Gothenburg, Göteborg, Sweden, and Lappeenranta Lahti University, Lappeenranta, Finland

The landscape of Software Engineering evolves rapidly amidst digital transformation and the ascendancy of AI, leading to profound shifts in the role and responsibilities of Software Engineers. This evolution encompasses both immediate changes, such as the adoption of Large Language Model-based approaches to coding, and deeper shifts driven by the profound societal and environmental impacts of technology. Despite the urgency, there persists a lag in adapting to these evolving roles. This roadmap article proposes 10 research challenges to develop a new generation of Software Engineers equipped to navigate the technical and social complexities as well as ethical considerations inherent in their evolving profession. Furthermore, the challenges target role definition, integration of AI, education transformation, standards evolution, and impact assessment to equip future Software Engineers to skillfully and responsibly handle the obstacles within their transforming discipline.

CCS Concepts: • **Software and its engineering** → **Software creation and management**; • **Social and professional topics** → **Computing education**; **Codes of ethics**;

Additional Key Words and Phrases: Sustainability, Responsibility, Roles, Ethics

ACM Reference format:
Stefanie Betz and Birgit Penzenstadler. 2025. With Great Power Comes Great Responsibility: The Role of Software Engineers. *ACM Trans. Softw. Eng. Methodol.* 34, 5, Article 136 (May 2025), 21 pages.
https://doi.org/10.1145/3715112

1 Introduction

The role of Software Engineers[1] changes. With digital transformation and the rise of AI, Software Engineers (as everyone else) are affected by short- and far-reaching changes, for example, by changing daily work practices (using LLMs for coding), but also a rise in complexity when developing new software due to the constantly changing requirements resulting in new features and the emergence of new technologies. Moreover, Software Engineers are not simply "executive assistants,"

[1]Based on Ian Sommerville we define Software Engineers as Engineers, who are concerned with all aspects of Software production [129].

Authors' Contact Information: Stefanie Betz (corresponding author), Department of Computer Science, Furtwangen University, Furtwangen, Germany and Lappeenranta-Lahti University of Technology, Lappeenranta, Finland; e-mail: stefanie.betz@lut.fi; Birgit Penzenstadler, Chalmers University of Technology, Göteborg, Sweden, University of Gothenburg, Göteborg, Sweden, and Lappeenranta Lahti University, Lappeenranta, Finland; e-mail: birgitp@chalmers.se.

ACM 1557-7392/2025/5-ART136
https://doi.org/10.1145/3715112

Fig. 1. Be the change, Brett Jordan, Unsplash, 2024.

even if they sometimes see themselves that way [19]. They work in all phases of the IT product and service development process. Starting from the early requirements, over coding and testing to the maintenance phase [26]. Moreover, nowadays sustainability issues demand responsible design decisions as IT products and services have short- and far-reaching impacts on societies and environment [15]. Thus, the role of Software Engineers changes and continues to do so (Figure 1). This means Software Engineers need to adapt (their role) both quickly and in the long run.

However, this is not happening to the degree necessary. Donald Gotterbarn already points out the lack of education for moral responsibility of Software Engineers back in 1995 [56] and becomes one of the champions in developing the codes of conduct in our field. Almost 30 years later we still primarily educate future professionals for a commercial market instead of responsible citizenship—while that is more urgent than ever because of software ruling our daily lives [14, 114, 132].

This lack of reflection in computing and computing education is also criticized by Becker in his recent book “Insolvent” [14]. Becker argues that the current (dominant) way of thinking in computing is not sufficient to address the grand challenges we face. He suggests reorienting design in computing to better align with the values of sustainability and justice [14]. Moreover, in [19] Betz et al. show in a small interview study that the current perception of Software Engineers regarding their role is still very classical, concentrating on hands-on tasks such as coding. In addition, the rapid roll-out of technological advancement using AI tools for Software Engineering changes the role of Software Engineers by partially or completely substituting tasks, which are subsequently replaced by higher-level tasks to review and verify [9, 86, 98].

To summarize, there is a need for research regarding current challenges in terms of complexity, sustainability, and responsibility, and how the role of the Software Engineer changes along with the ever-increasing impact of their profession on everyday life. However, to the best of our knowledge, there is not much work to be found on the role of Software Engineers and their perception of it. This gap is also visible in the smaller number of ethnographic studies (compared to other research methods) in Software Engineering [125, 141], a research method that serves well to investigate human aspects. In the words of Gotterbarn: “Professionalism maintains a proactive posture” [58, p. 4].

This article calls for evolving Software Engineers for their manifold role and turning them into stewards of their profession beyond commercial practice. Our *contribution* entails (1) a *roadmap* with 10 research challenges investigating the role of Software Engineers, (2) a draft of a suitable research approach for each, and (3) a theoretical perspective for each of these challenges.

2 Background and Related Work

Roles play a crucial part in our lives [142]. The concept of roles has long been utilized in fields like sociology, management, behavioral science, and psychology. They serve as effective tools for modeling the authority, responsibilities, tasks, and interactions tied to roles within organizations [142]. There exist many different definitions of roles [6]. However, most prominent is the concept of roles in sociology. There, the understanding and idea of the concept of role is to categorize individuals into different "roles." Thus, in our article, we use the following definition of "roles" to form a basis of understanding: "A position containing a set of socially defined attributes and expectations that determine appropriate behavior for an individual or group based on their status in relation to other people or groups" [16].

To comprehend how roles impact individuals' lives and organizations, sociologists, behavioral scientists, and psychologists formulate role theories. These theories elucidate the influence of roles on individual behavior and how such behavior is evaluated by others concerning existing roles [6]. These roles shape both the behavior and self-perception of individuals, as well as how others interpret their actions [6]. Furthermore, as stated in [102]: "role expectations include both activities and qualities: a teacher may be expected to not only deliver lectures, assign homework, and prepare examinations but also to be dedicated, concerned, honest, and responsible." Transferring role theory to Software Engineers shows that they also "play" a certain role that has been socially constructed over time. If the role changes, the result of their work may also change. In this case, this means that the software system as the result of Software Engineers' work may also change.

Goffman [55] argues that roles form and then are difficult to discard. Once a role emerges, it solidifies over time and it sometimes requires a great deal of work to change it [55]. The proof-of-concept document of the Guide to the **Software Engineering Body of Knowledge (SWEBOK)**[2] is made available in 1998—now more than 25 years ago. While SWEBOK does not explicitly define the role of a Software Engineer, it defines tasks and activities and it states "A Software Engineer displays professionalism notably through adherence to codes of ethics and professional conduct" [26, Chapter 11, p. 2].

However, Garousi et al. [51] identify knowledge gaps in software maintenance, software configuration management, and testing in SWEBOK. To understand the quality attributes of a Software Engineer, Li et al. [84] perform 59 semi-structured interviews, spanning 13 Microsoft divisions, including several interviews with architect-level engineers with over 25 years of experience. The contribution of their effort is a thorough, specific, and contextual understanding of Software Engineering expertise, as viewed by expert Software Engineers. However, they also argue that this study is just a beginning, that more studies are needed to get an understanding of what the role of Software Engineers looks like and that "our research community can begin to understand software engineering not just as a purely technical discipline of tools and processes, but a sociotechnical one, with individual human contributors and their collaborations fueling software progress" [84, p. 709].

More recently, Betz et al [19] investigate how Software Engineers perceive their role, which attributes they ascribe to themselves and their awareness with regard to sustainability and they find out that Software Engineers seem to see themselves mainly as an executing force as opposed to a directing one. In summary, the Software Engineer's role, their activities and qualities (responsibilities) should and need to evolve in the light of the current developments.

2.1 The Significance of AI

At present, our society depends on software systems for global communication and the operation of essential utilities. In general, these systems prioritize functionality and quality characteristics like usability, availability, safety, and security [95, 121]. However, they must advance beyond these and consider ethical issues, such as social justice, transparency, or diversity.

[2] https://www.ieee.org/about/ieee-india/ieee-computer-society-india/swebok.html

This becomes even more urgent with the rise of AI. It is imperative to conduct research and enhance awareness concerning ethical considerations, including "transparency, accountability, confidentiality, autonomy, trust, and fairness" [80, 88]. Also, for example with regard to smart grids there are ethical issues which should be taken into account when designing such software. One of the issues evolves around the customer and the software system (trust, transparency, . . . see [81]) and another around the topic of access to micro grids and social inequality [103, 133].

In their very recent paper, Pant et al. [105] investigate ethics in AI from an IT practitioners point of view. They develop a taxonomy to help AI practitioners in identifying and understanding AI ethics. Moreover, they present recommendations for IT industry and future research. For example, they advocate examining different roles and their interactions in the development and use of AI systems.

The first paper to address AI and Software Engineering is already from 1988, and it look at attempts to apply AI techniques to Software Engineering problems [12]. This technology has increased the scope of our research tremendously, and with that also increased the responsibility of Software Engineers. Currently, there is a large amount of work exploring the uses of AI and less work taking a critical perspective on responsibility around AI within SE. Feldt et al. [47] provide an AI in SE Application Levels taxonomy. The taxonomy has three facets allowing its users to classify, the point of application (process, product, and runtime), the type of AI technology (based, initially, on the five tribes proposed by Domingos [41]), and the level of automation of the applied technology (inspired by Sheridan–Verplanck's 10 levels of automation [127]). As the technology evolves at a tremendous speed, more and more approaches are being developed to support, automate and partially substitute Software Engineering activities, for example, in Requirements Engineering [86, 98], in Software Architecture [45, 68], and in repository-level coding tasks [9].

To summarize, this is changing the daily activities, responsibilities, and role of Software Engineers and needs to be investigated by Software Engineering researchers.

2.2 Responsibility and Ethics

In general, there are many different keywords (and work) to be found in the area of responsibility, ethics, and Software Engineering: accountability, ethics, ethics-aware and ethical-driven, responsibility, values, and codes. Following, we look at selected exemplary work in each of these.

David Gotterbarn was one of the first to take a strong stance for the moral responsibility of Software Engineers [56, p. 54], noting "a surprising lacuna in Software Engineering education, one of whose primary foci is the preparation of software professionals for the commercial market." However, it seems there is still a lack of research in the area of practical ethics and Software Engineering. In their recent mapping study (2023) Marebane et al. [92] investigated studies on the ethical responsibility of Software Engineers. They only found 14 relevant papers, most of them being philosophical studies and no validation or experience papers.

With regard to accountability, Schneider and Betz [120] wrote that Software Engineering needs to become accountable for sustainability. In order to do so they combine a social scientific sustainability transformation model with the analysis of accountability for sustainability in current Software Engineering practices. They argue that for Software Engineers to become accountable for sustainability a mind shift needs to be established [120]. This mind shift naturally affects also the roles of Software Engineers. Aydemir and Dalpiaz [8] present in their paper a roadmap for an analytical framework for ethics-aware Software Engineering. In ethics-aware Software Engineering the idea is that the ethical values of stakeholders (including Software Engineers and users) are taken into account during the software requirements engineering and development phase. The framework identifies the subject (what), the affected object (who) and the relevant value (what). Although, this framework seems quite helpful, it does not touch upon the issue of changing one's role. In their philosophical paper, Génova et al. [52] argue that as a practical guide for ethically responsible Software Engineering one should stick to a moderate deontologism taking into account rules and consequences when assessing the ethical impact of Software Engineering practices. They argue that this is due to the need of assessing possible short- and long-term impacts of design decisions (consequences) while taking into account "absolute human values" (rules). However, they do not provide methods and tools for the assessment. Lurie and Mark [87] propose a practical ethical framework for Software Engineers connecting Software Engineers' ethical responsibilities directly to practical activities to overcome the dichotomy between professional skills and ethical skills. They call the framework Ethical-Driven Software Development, as an approach to software development. The goal of the framework is not to affect the outcome of the Software Engineering process but to raise Software Engineers' awareness regarding the "ethical ramifications of certain engineering decisions within the process" [87, p. 1]. Awareness is definitive a first step, but as following steps we need assessment of impacts on ethical issues. In their paper from 2022 Mitchell et al. argue based on an interview study that there exists a "lack of tool and process support for systematic ethical deliberation at most stages of the software lifecycle" [101, p. 1].

Values and ethics are not the same. Ethics are trying to find measures for just, good, and desirable action and values can be one of these three things or al all three: "(a) biological needs, (b) interactional requirements for interpersonal coordination, and (c) societal demands for group welfare and survival" [123, p. 1]. Thus, values do not lead us, per definition, to just, good, and desirable actions.

Nevertheless, research exists on values within Software Engineering. Several studies adopt a human-values perspective, such as [67] which gave recommendations on how to develop "values-conscious software," [48] they created a research approach that facilitates a systematic method for the elicitation, analysis, and comprehension of values, and [5] which integrated ethical values in software architecture design. However, more efforts are still needed to systematically incorporate human values into Software Engineers' daily work practices.

In summary, software systems ought to adhere to ethical guidelines for Software Engineering such as those outlined in the ACM Code of Ethics [57]. However, although the ACM Code of Ethics for Software Engineering has been around for quite some time it seems that it has no influence on software-related ethical decisions [96]. As Karim et al. [71] state despite a sufficient large number of research studies on Software Engineering Codes of Ethics operationalizing and implementing those codes in the daily work practices is quite difficult and not yet solved. We argue in order to do so we need to change the Software Engineering roles and their responsibilities, tasks and activities.

2.3 Learning and Education

Mary Shaw's roadmap [126] proposes challenges for educators of software developers to help identify aspirations for Software Engineering education. As Software Engineering educators, we have been working on a clarification of the roles involved in software development and Shaw points out that challenges are: appropriate credentialing for those roles; an improved treatment of engineering issues; faster response of educational content to changes in technology and fundamental understanding; and better use of information technology in our own education and training. In this line of research, we are contributing to a further, and updated, clarification of the role of a Software Engineer as well as the continued change that comes with this profession and its tasks.

The challenges of Software Engineering education presented by Jazayeri [70] in 2004 still hold to a large degree since they are inherent to the discipline. In addition, we have to integrate relevant topics like sustainability, responsibility, inclusion, and ethics to a larger extent into SE teaching.

In [113] Peters et al. perform a systematic review of the literature on sustainability in computing education. From a set of 572 publications extracted from 6 large digital libraries plus snowballing, the authors distill and analyze 89 relevant primary studies. They investigate (i) conceptions of sustainability, computing, and education; (ii) implementations of sustainability in computing education; and (iii) research on sustainability in computing education. The result is a framework capturing learning objectives and outcomes as well as pedagogical methods for sustainability in computing education.

A central contribution in educating for sustainability is the key competencies framework [27] that lays out systems thinking competency, strategic competency, values-thinking (or interpersonal) competency, future (or anticipatory) competency, and implementation (or problem-solving, normative) competency. As part of values thinking, it is considered important to integrate ethical education into SE curricula [53, 61] as well as the notion of responsibility [56, 119] to establish the actual application of a code of ethics [57]. As part of future thinking competency, it is relevant to include technology forecasting [7, 33] and technology assessment [46, 62]. Furthermore, systems thinking needs a place in software engineering education [19], including teaching the limits of computational thinking [14]. Finally, all of these competencies should be anchored in SWEBOK [26], where the term "sustainability" currently shows up in two instances in the 2024 public review draft: (i) as architecture concern and (ii) as economic concern. Both the environmental and the social dimension are still missing.

With regard to learning there is work on lifelong learning as Software Engineers from Chernenko [32]. Also, there is work on individual happiness of Software Engineers [59, 109] and on team happiness [60] and how this impacts performance.

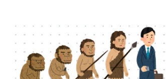
Some of these works are used as a theoretical perspective for challenges 3.3 *Integration of Responsibility Considerations*, 3.4 *Promoting Self-Reflection and Lifelong Learning in Software Engineering*, 3.6 *Defining the Ideal Role*, and 3.8 *Educational Curricula* described in the following section.

3 Research Challenges

The overall objective of this research roadmap is to shed light on how a Software Engineer's role, their activities, and qualities (responsibilities) should and need to evolve in the light of the current developments. This objective is elaborated in the following central questions:

- What is the role of a Software Engineer now?
- What role do we want and need to have, given that we build the systems that run the world?
- How can we steer the role change in terms of technology support, sustainability, and ethical responsibility?
- How do we get the new role definition into education and practice?
- How do we keep up with the ever-evolving role and the need for continued self-reflection?

In the following, we present and explain the 10 research challenges that help answer the above questions. For each *challenge* (illustrated with an icon), we first present the challenge and the corresponding research questions, then provide a first draft of a *research approach*, followed by a *theoretical perspective* pointing to relevant research work.

3.1 Understanding the Evolving Role of Software Engineers 

Challenge. What are the current and emerging tasks and responsibilities of Software Engineers beyond traditional tasks like coding and testing? This includes the question of how AI changes these tasks and responsibilities in the short and long term given current social developments. It furthermore encompasses what the new tasks of Software Engineers are if an AI system does the coding. The impact on Software Engineers' self-perception should be investigated if one of

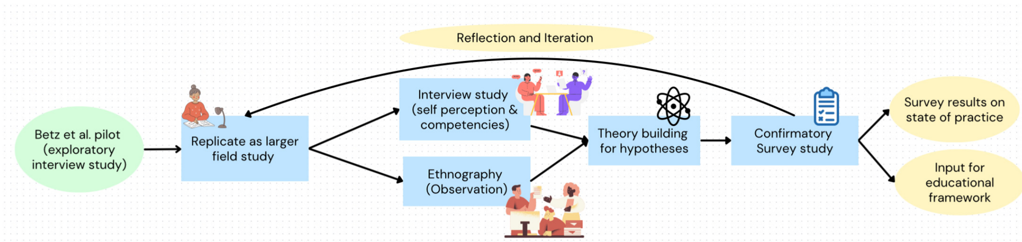


Fig. 2. Understanding the evolving role of a Software Engineer.

their major tasks (coding) is not carried out by themselves anymore. We also ask how new tasks and responsibilities come into play, such as ethical checks of the system under development and sustainability issues, to face current and future societal challenges.

Research Approach. We propose to conduct qualitative studies (e.g., interviews, ethnographic studies) with Software Engineers, employers, and industry experts to explore these evolving roles. Qualitative studies allow researchers to investigate emerging practices in a real-life context [140]. The initial phase involves conducting an ethnographic study to examine how Software Engineers are currently managing the evolving roles and responsibilities that extend beyond traditional activities such as coding and testing. It is suitable to employ ethnographic research, as it is most appropriate to examine the behaviors of a culture-sharing group (or individuals), such as Software Engineers [38]. Furthermore, conducting an ethnographic study is recommended because these provide the best opportunity to gather current, reliable, comprehensive, and precise data to address relevant research questions [134]. Betz et al. [19] find that Software Engineers still see themselves mainly as coders that implement as they are being asked to without questioning the specifications too much. Since this study was based on a small sample, the first step in the vision at hand is to replicate the study with a larger number of interviewees. This larger study could, for example, use observations and interviews and will give us the hypotheses upon which to base our confirmatory study; see Figure 2.

Theoretical Perspective. According to Stol's ABC [131], the right place to study the current role of Software Engineers for the purpose of studying actors and their behavior in natural contexts is the "jungle": "Natural setting that exists before the researcher enters it. Minimal intrusion of the setting to not disturb realism, only to facilitate data collection" [131, p. 13]. Sharp's review of ethnographic studies [125] reveals that there used to be a small set of them within Software Engineering, most likely due to the fact that pure ethnography had been assigned lower value in Software Engineering before also offering the evaluation of an intervention (field experiment)—this is changing in recent years [141].

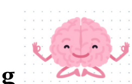
3.2 Mapping the Societal Impact of Software Engineering

Challenge. How does the role of Software Engineers influence broader societal issues, such as sustainability and ethics? We propose investigating to evidence direct, indirect, and systemic effects of Software Engineers' practices to showcase the impacts and responsibilities they have in their decisions. This includes how to identify possible social, individual, and environmental effects of software systems on their surroundings. Furthermore, this encompasses how to measure and quantify such effects as well as how to deal with the uncertainty ingrained in long-term effects.

Research Approach. We propose to investigate this with cross-disciplinary studies [85] combining computer science with environmental sciences (also known as environmental informatics [104]) and ethical studies to assess the societal impact of Software Engineering practices [95]. When talking about societal impacts, one should include scenario analysis, a common method for understanding possible futures [78], and perform validation experiments on these scenarios to identify possible effects and chains-of-effects. To bring these results into relation with each other, we advocate for using the **Soft Systems Methodology (SSM)**[3] as a method for systems thinking analysis to identify and analyze positive and negative feedback loops.

Theoretical Perspective. Use frameworks of system thinking [31] to identify and assess possible direct, indirect, and systemic effects, as well as chains of effects [20]. In addition, leverage points as defined by Meadows [97] and applied to Software Engineering by Penzenstadler et al. [110] can be used to look at the impacts of Software Engineering practices. Scenario analysis could be used to assess dynamic aspects of Software Engineering technologies still under development [116].

3.3 Integration of Responsibility Considerations in Software Engineering



Challenge. How can ethical considerations, including social implications, be systematically integrated into the evolving Software Engineering role? We wonder how a code of ethics can be applied in the day-to-day work environment settings. This includes how to address software design decisions to minimize the potential misuse of software systems. We need to understand how the integration of AI facilitates or hampers such application. And whether we need a different set of responsibility considerations when working on different types of systems (e.g., a computer game, a procurement system or an aircraft flight control system). The latter could also be seen as one set of considerations with more or less emphasis of different aspects (e.g., gender balance, accessibility, and individual freedom). We propose to investigate how to design and evaluate an approach to support responsibility engineering in daily work practice for Software Engineers that they integrate bottom-up in to their routines.

Research Approach. Answering these questions requires assessing existing ethical and responsibility frameworks for how to integrate more closely with a Software Engineer's activities. The first step of that analysis is to differentiate the most relevant frameworks to assess and learn from, for example, the ACM Code of Ethics [57] as well as value-sensitive design [48, 50] and more task-specific frameworks, as for requirements engineering [21]. Based on an understanding of the current view of their (perceived lack of) responsibility, we envision to roll out a larger field experiment [131] in order to investigate how Software Engineers deal with cases presented that have ethical concerns. On this foundation, it is necessary to further develop and test frameworks or guidelines to incorporate ethics into software development processes.

Theoretical Perspective. As stated, one of the first to look into responsibility of Software Engineers was Gotterbarn [56] who then proceeded to initiate the ACM Code of Ethics [57]. Later on, Génova et al. [53] as well as Grubb [61] emphasized the need for ethical education in Software Engineering. The more technical perspective of responsibility for quality has been researched by Schieferdecker [119].

[3] SSM is an approach for tackling problematical, messy situations of all kinds. It is an action-oriented process of inquiry into problematic situations in which users learn their way from analyzing the situation to taking action to improve it. Learning emerges through an organized process in which the situation is explored using a set of purposeful action models (each built to encapsulate a single worldview) as intellectual devices or tools, to inform and structure the discussion about a situation and how it could be improved [31].

3.4 Promoting Self-Reflection and Lifelong Learning in Software Engineering 

Challenge. What methods can encourage continuous self-reflection and lifelong learning among Software Engineers? This includes how Software Engineers engage with critical reflection as well as aspects like psychological safety in teams. Furthermore, it poses the question of how emotions impact communication both on an individual and a team level. We also propose to investigate how well-being influences individual and team performance. That leads to the question of how to motivate Software Engineers to work on their self-perception. Overall, we are curious how Software Engineers evolve along with their evolving role.

Research Approach. We propose to evaluate the practices of debrief meetings and reflective journaling. Reflective journaling is the most common technique for personal reflection, allowing the connection between thought and the world to be more profound and meaningful, which in turn expands inner perspectives [37]. Subsequently, one can design and evaluate tools or programs (e.g., workshops, online platforms) that promote reflective practice and ongoing education. One starting point to promote self-reflection and life-long learning could be to iteratively improve understanding of Software Engineering's role and the responsibility that comes with it. Explained in stages that use the terminology of the revised Bloom taxonomy [49], we propose to approach a continued discourse and self-reflection of one's work practice as described in Table 1. We see the six levels of the revised Bloom taxonomy and an explanation of their basic understanding in the left column, then an example activity in the middle column, and more details about how this could be carried out within a Software Engineering company in the right column.

Theoretical Perspective. Two learning approaches to use here are Gibbs' reflective learning approach [54] and Kolb's learning style [76]. Behavioral Software Engineering [82] has developed over the last 10 years as an area of study, plus we get to look at lifelong learning as Software Engineers [32]. Furthermore, there are specific aspects like psychological safety in teams and its impact on secure Software Engineering [60], happiness of developers [59] and how to support happiness in organizations [24], the well-being of developers [109, 112], and finding flow in community [108].

3.5 Assessing the Impact of Non-Technical Skills in Software Engineering 

Challenge. What non-technical skills (e.g., communication, teamwork, problem-solving, change management) are increasingly critical for Software Engineers, and how can they be cultivated? This includes how to better integrate target audiences like senior citizens who may struggle with the increasing reliance on software systems for their health as well as their civic responsibilities and administrative processes, for example, taxes and municipal services. We also are curious how to better integrate target audiences that may not speak the local language nor are familiar with how public administration in a given country works, for example, migrants looking for how to get a job or extend a residence permit. It is worth pondering what non-technical skills are essential for Software Engineering in disaster relief, and how they can be cultivated to address the growing need for crisis management systems in the face of increasing (environmental) catastrophes such as floods and wildfires, given the high-pressure, emotionally charged, and fast-paced nature of these situations.

Research Approach. We propose a mixed-methods approach that combines both assessments of Software Engineers' capabilities and evaluations of target audiences' needs [3, 66]. This includes surveys and skill assessments to identify and validate the importance of these skills. Surveys are

Table 1. Continued Discourse and Self Reflection of One's Work Practice

Bloom's	**Activity**	**Details**
Remember: Recall facts and basic concepts	Training for sustainability awareness	Seminar during onboarding plus yearly update
Understand: Explain ideas or concepts	Exemplary case study discussions	Solidify concepts in quarterly training
Apply: Use information in new situations	Weekly reviews with personal reflections	Ensure cognizance of ongoing themes in self-reflection, for example, report in daily stand-up meetings if ethical issues occur
Analyze: Draw connections amongst ideas	Monthly reviews of personal reflections, for example, as part of the sprint retrospective	Aggregate higher level reflection to identify potential patterns
Evaluate: Justify a stand or decision	Reviews for ethical issues in company using focus group meetings	Take notes to collect a database of issues and how they are investigated, tracked and decided, for example, include ethical issues in user stories
Create: Produce new or original work	Guidelines for individual company	Develop a set of tailored guidelines for one's company

used to gather evidence from a given larger population [72] and skill assessment helps to better understand what skills are missing to what degree [28].

Additionally, we consider it vital to conduct comparative ethnographic studies [128] in classical Software Engineering situations and during specific, critical circumstances, for example, crisis or war, to investigate what non-technical skills are needed, and if and how they differ [79]. On this foundation, exemplary guidelines and processes can be developed on how to develop software under specific, critical circumstances.

Theoretical Perspective. There is a plethora of relevant work building the foundation for this topic, from stakeholder communication [77] and communication science [18] to participatory design [122] as well as stakeholder development especially in sustainability contexts [17]. Furthermore, we draw from developing software systems in crisis [40, 83] for disaster preparedness that commences with factors that motivate people to prepare, progresses through the formation of intentions, and culminates in decisions to prepare [107]. Finally, occupational health [130] and senior citizen science [11, 75, 117] are avenues for exploration.

3.6 Defining the Ideal Role of Future Software Engineers

Challenge. What competencies, tasks, and responsibilities should define the Software Engineer of the future, considering technological advancements and societal needs? This includes the question of what a Software Engineer needs to learn and to know in order to best fulfill their future role and to step up to their future tasks. Given recent developments the topic also begs to investigate how this reflects in the increased integration of AI.

Table 2. Required Competencies, Methods and Technologies for a Changing Role

Competency	**Method (Refines Competency)**	**Technology (Refines Method)**
Long-term thinking [14]	Systems thinking and critical thinking [14]	Soft systems methodology [31]
Complexity theory [94]	Consideration of and working in and between different abstraction levels	Agent based modelling [91]
Environmental externalities and impacts [64]	Life cycle analysis [35], sustainability assessments [23]	Impact pathways [43]
Diversity Equity Inclusion	Ethnography [125]	Data science understanding, bridging the digital divide
Ethical concerns [52]	Ethics as a skill [69]	Modelling Ethics for SEs [4]

Research Approach. To research the above questions, we propose Delphi studies. Delphi studies are instruments of prognostics, which takes place among experts from various specialist areas [137]. It is assumed that experts possess extensive knowledge in their respective domains, enabling them to provide well-founded estimates of potential future developments. We find it important to include AI experts on the current and future evolution of the AI coding generators. We include polling sustainability and ethical software engineering experts in understanding possible new tasks and responsibilities using, for example, focus groups [2].

To identify the required new activities, tasks and responsibilities defining the new role of Software Engineers, we present here three abstraction levels: Competencies, methods (refining competencies) and technologies (refining methods). They help us think through this improved understanding of roles (see Table 2).

For example, in order to consider long-term thinking during IT product and service development, one should apply systems thinking and critical thinking using, for example, the SSM. In this case the Software Engineering's role would be extended with regard to responsibilities by applying long-term thinking to raise awareness regarding sustainability issues as well as activities and tasks supporting it (e.g., a soft systems modelling exercise).

This is an initial framework, where the different wording, methods, and tools are only a proof of concept and can be discussed. We plan to further investigate and extend this based on a literature study and the ethnographic study described above. We started with these competences as long-term thinking is crucial for understanding and assessing the current Grand Challenges faced by humanity. We added Complexity Theory as Software Systems become more and more complex also given the current AI trend. Environmental externalities have been included as climate change is affecting all of us, and IT Products and Services are one way to minimize negative impacts of it. Finally, we included diversity, equity and inclusion to support the integration and leverage diverse perspectives that promote innovation and collaboration and ethical concerns to address responsible citizenship.

Theoretical Perspective. We recommend starting with technology forecasting [7, 33] and technology assessment [62, 135] including its specific angle for sustainability [46]. Additionally, it is good to utilize frameworks from occupational sociology to define and categorize the role and its tasks and responsibilities [1] and standards such as SWEBOK [26].

3.7 Navigating Role Transformation in Software Engineering

Challenge. What mechanisms and strategies can be employed to guide the evolution of the Software Engineer's role, particularly towards more ethical and sustainable practices? We are curious what the legislative frameworks and policies are that can support this evolution. We need to understand their current status, whether there is enough support, and whether there is a gap that needs to be filled *via* new regulations or alternative inventions. We furthermore ask how the differences between the regulations in different countries affect Software Engineers and Software Engineering. This challenge is about top-down support of the role transformation by legal boundaries and steering.

Research Approach. To answer this challenge, one can look into the area of risk management and Software Engineering and see what can be adapted [118]. Legislation to regulate engineering was introduced in response to collapsing bridges and similar failures. For example, Canadian provinces as well as most US States have regulatory bodies that ensure that (i) qualified engineers are easy to identify; (ii) (licensed) engineers remain qualified; and (iii) regulatory bodies remove the license of engineers who violate laws or codes of practice [106]. The main objective in this challenge is to evaluate whether such a route would be more beneficial or more detrimental to Software Engineering. For this, we need policy analysis as well as the development of legal frameworks for guiding professional practice, incorporating stakeholder perspectives [66]. It may be worth mentioning that there are professional bodies for Software Engineering, for example, ACM, BCS, The Chartered Institute for IT [89], but they are weak because there is no legal nor commercial requirement for professional registration in order to work in Software Engineering.

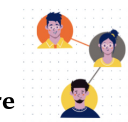
Theoretical Perspective. Already in 2002, Parnas made a case for licensing Software Engineers akin to Canadian engineers [106]. Of course, there are also good reasons against such a licensing, for example, reduced interdisciplinarity and freedom, and the drawbacks of a single professional model [93]. Additionally, one needs to be guarded against possible self-interest of accreditation boards [93]. Examination of the evolution of historical roles in other professions as case studies, as for example in [90]. This needs to incorporate occupational law as well as public policy on a national and international level.

3.8 Educational Curricula and the Future of Software Engineering

Challenge. How should educational curricula adapt to prepare Software Engineers for their evolving roles? This includes which new skills need to be included, for example sustainability and ethics to name two responsibilities. Furthermore, we ask how and to what extent is teaching of coding skills needed if the technology advancements make them potentially obsolete.

Research Approach. For this topic, we propose a comparative analysis of existing curricula in Software Engineering and related areas such as Computer Science versus emerging needs [113]. On that foundation, software engineering education researchers can develop an adapted curriculum with the possibility to tailor it to different countries and needs (such as a focus on energy consumption of AI systems versus the need for responsible systems for elderly care) [22, 100]. Those would be followed by pilot programs to test the new educational models. This would ideally be investigated simultaneously in different countries since curricula can vary [25]. In such a research network, a joint development of adapted curricula could be defined such that roll-out of pilot studies is orchestrated within a reasonably short evaluation period. This is necessary to plan for early on since curriculum changes tend to be slow and otherwise run way beyond the time of an individual Ph.D. Subsequently, SWEBOK [26] and the ACM SE curriculum need to be extended.

Theoretical Perspective. Pedagogy and higher education provide the backdrop for this topic, including but not limited to standard Software Engineering curricula (e.g., [26]), sustainability education [113], computational and systems thinking [19], limits of computational thinking [14], and ethics education in Software Engineering [52, 61].



3.9 Role of Professional Organizations in Shaping the Software Engineering Profession

Challenge. How can professional organizations influence the definition and adoption of new roles for Software Engineers? We need to understand whether the existing professional standards are sufficient to fulfill this task or whether they need to evolve. If so, how to integrate new topics like sustainability, responsibility, and inclusion into standards and certification.

Research Approach. This topic asks for case studies of successful role transformations facilitated by professional bodies, development of strategic recommendations, and lessons learned [99]. It also requires evolution of professional certifications, for example the extension of the IREB body of knowledge [111] or Scrum certifications, with modules around sustainability, responsibility, inclusion, and ethics. We advocate for applying design science research to investigate and evolve existing professional standards [10], because design science research generates knowledge and scientific evidence by systematically developing an artifact to solve human problems in a particular context [20, 138].

Theoretical Perspective. We integrate knowledge from organizational change management [30] and look at knowledge transfer [29], for example, IREB Requirements Engineering Fundamentals [115], SIGSOFT CARES [124], Scrum Master [13], Six Sigma [136], CMMI [34], and others.

3.10 Measuring the Effectiveness of Role Evolution in Software Engineering

Challenge. How can we measure the success and impact of changes in the role of Software Engineers? How do we identify success and impact of the developments in the workforce? This includes investigating how we can show the effects of a positive evolution of the profession of Software Engineering in terms of societal impact.

Research Approach. For this topic, we advocate to (i) develop metrics and an evaluation framework that track professional development in general (what the changes are within tasks and responsibilities) [36], (ii) develop metrics and an evaluation framework on an individual level for job satisfaction [74], and (iii) develop metrics and a framework to measure societal contributions of Software Engineers as their roles evolve. Societal contributions of software engineers have not yet been measured [63]. All three of these can be carried out as individual Ph.D. projects.

Theoretical Perspective. Potential starting points for this challenge include evolutionary stages for personal development [73], personal development and personality models [39], the concept of emotional intelligence [44], as well as spiral dynamics [139] and integral dynamics [42].

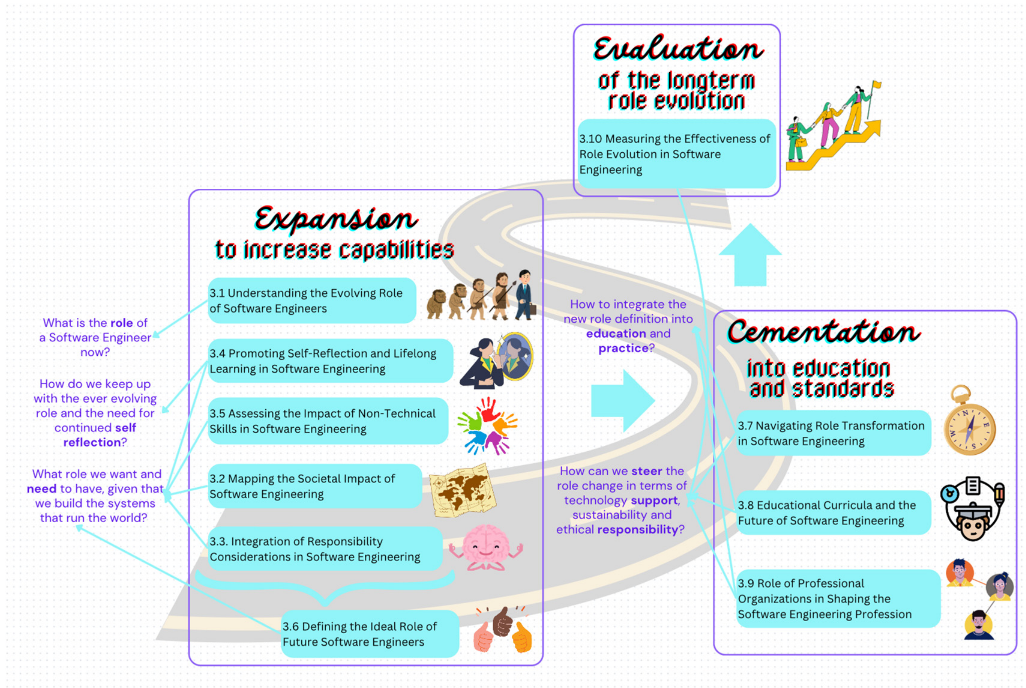


Fig. 3. Research roadmap for the role of a Software Engineer.

4 Research Roadmap for the Role of a Software Engineer

These 10 challenges aim to foster a deeper understanding and proactive shaping of the Software Engineering profession in response to technological, societal, and ethical developments. We present a roadmap for a line of research investigating the understanding of the role of Software Engineers *via* a wide set of approaches and methods as listed in the research challenges above. The challenges can be mapped into the research roadmap as depicted in Figure 3. The research roadmap consists of the three phases: Expansion, Cementation, and Evaluation. The first one *Expansion to increase capabilities* focuses on exploration and capability development, it builds the foundation for the second phase *Cementation into education and standards*. Once these capabilities have been explored, the next step is to formalize and institutionalize the knowledge into standard practices, educational curricula, and technical guidelines. Finally, the last phase *Evaluation of the long-term role evolution* focuses on assessing the long-term impact of the research field. Subsequently, we provide an elaboration on the execution of the three phases. In particular, we incorporate descriptions of the implementation of the challenges in each of the three phases.

In the first phase *Expansion* the challenges 3.1–3.6 are to be tackled to evolve the current role of Software Engineers to answer the call to both shifts, immediate ones such as the adoption of AI based approaches in coding, and long-term ones driven by the heavy societal and environmental impacts of technology. In this phase, the first five challenges 3.1 *Evolving Role of Software Engineers* to 3.5 *Impact of Non-Technical Skills* are all to be investigated in parallel. Additionally, they are feeding into the last challenge in this phase 3.6 *Defining the Ideal Role of Future Software Engineers*.

The first phase of our research roadmap provides the necessary foundation for addressing three of our research questions:

—What is the role of a Software Engineer now?
—How do we keep up with the ever-evolving role and the need for continued self-reflection?
—What role we want and need to have, given that we build the systems that run the world?

In the second phase *Cementation* three further challenges on 3.7 *Navigating Role Transformation*, 3.8 *Educational Curricula*, and 3.9 *Role of Professional Organizations* are investigating how to ensure that the evolution of new roles and their associated new tasks and responsibilities are going to be implemented into the Software Engineer's daily work. These challenges should be assessed as the first phase nears completion, given that their outcomes inform the next phase. The three challenges can and should be tackled in parallel.

Finally, the last phase *Evaluation* and its challenge 3.10 *Measuring the Effectiveness of Role Evolution* is meant to measure and evaluate the changes and their effectiveness. Naturally, this challenge is tackled after finishing the first two phases.

Addressing the research challenges in phase two and three helps answering the remaining two research questions, see Figure 3:

—How to roll out and integrate the new role definition into education and practice?
—How can we steer the role change in terms of technology support, sustainability, and ethical responsibility?

5 Conclusion

In this article, we delineate 10 research challenges pertaining to the evolving role and responsibilities of a Software Engineer, each accompanied by critical inquiries, methodological approaches, and theoretical foundations. We conclude with an illustrative overview (Figure 3). We assert that this framework effectively serves as a guiding tool for future scholarly research in this domain.

However, inherent to research challenges are several limitations: First, the 10 challenges presented here are not all at the same level of abstraction. For example, challenge 3.10 *Measuring the effectiveness of role evolution in Software Engineering*, is at a much higher level of abstraction than for example challenge 3.6 *Defining the ideal role of future Software Engineers*. Second, we might have missed a challenge. Third, of course, the challenges have dependencies in between them (see Figure 3) as well as some overlaps and/or lack of sharp scope boundaries.

Nevertheless, we argue that these challenges urgently need to be addressed by Software Engineering researchers in these unique times with fast changing technologies, the rise of AI as well as grand societal challenges.

Acknowledgments

To the Sun, thank you for coming back in Spring. We thank Dr. Ian Brooks for providing helpful feedback on an earlier version of this manuscript as well as the anonymous reviewers, especially reviewer 1, for constructive criticism and suggestions for improvement.

References

[1] Andrew Abbott. 1993. The sociology of work and occupations. *Annual Review of Sociology* 19, 1 (1993), 187–209.
[2] Ivana Acocella. 2012. The focus groups in social research: Advantages and disadvantages. *Quality & Quantity* 46 (2012), 1125–1136.
[3] Deniz Akdur. 2022. Analysis of software engineering skills gap in the industry. *ACM Transactions on Computing Education* 23, 1 (Dec. 2022), Article 16, 28 pages. DOI: https://doi.org/10.1145/3567837
[4] Sabah Al-Fedaghi. 2018. Thinging ethics for software engineers. arXiv:1810.02685. Retrieved from http://arxiv.org/abs/1810.02685

[5] Razieh Alidoosti, Patricia Lago, Eltjo Poort, Maryam Razavian, and Antony Tang. 2022. Incorporating ethical values into software architecture design practices. In *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 124–127.

[6] Aaron H. Anglin, Paula A. Kincaid, Jeremy C. Short, and David G. Allen. 2022. Role theory perspectives: Past, present, and future applications of role theories in management research. *Journal of Management* 48, 6 (2022), 1469–1502.

[7] J. Scott Armstrong. 2001. *Selecting Forecasting Methods*. Springer.

[8] Fatma Başak Aydemir and Fabiano Dalpiaz. 2018. A roadmap for ethics-aware software engineering. In *Proceedings of the International Workshop on Software Fairness*, 15–21.

[9] Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Vageesh D. C., Arun Iyer, Suresh Parthasarathy, Sriram Rajamani, B. Ashok, and Shashank Shet. 2024. CodePlan: Repository-level coding using LLMs and planning. *Proceedings of the ACM on Software Engineering* 1, FSE (July 2024), Article 31, 24 pages. DOI: https://doi.org/10.1145/3643757

[10] Béatrix Barafort, Anup Shrestha, Stéphane Cortina, and Alain Renault. 2018. A software artefact to support standard-based process assessment: Evolution of the TIPA® framework in a design science research project. *Computer Standards & Interfaces* 60 (2018), 37–47.

[11] Helen Barrie, Veronica Soebarto, Jarrod Lange, Fidelma Mc Corry-Breen, and Lauren Walker. 2019. Using citizen science to explore neighbourhood influences on ageing well: Pilot project. In *Healthcare*, Vol. 7. MDPI, 126.

[12] David Barstow. 1988. Artificial intelligence and software engineering. In *Exploring Artificial Intelligence*. Elsevier, 641–670.

[13] Julian M. Bass. 2014. Scrum master activities: Process tailoring in large enterprise projects. In *2014 IEEE 9th International Conference on Global Software Engineering*. IEEE, 6–15.

[14] Christoph Becker. 2023. *Insolvent: How to Reorient Computing for Just Sustainability*. MIT Press.

[15] Christoph Becker, Stefanie Betz, Ruzanna Chitchyan, Leticia Duboc, Steve M. Easterbrook, Birgit Penzenstadler, Norbet Seyff, and Colin C. Venters. 2015. Requirements: The key to sustainability. *IEEE Software* 33, 1 (2015), 56–65.

[16] Kenton Bell. n.d. Role definition. Retrieved from https://sociologydictionary.org/role/#definition_of_role

[17] Simon Bell, Stephen Morse, and Rupesh A. Shah. 2012. Understanding stakeholder participation in research as part of sustainable development. *Journal of Environmental Management* 101 (2012), 13–22.

[18] Charles R. Berger, Michael E. Roloff, and David R. Ewoldsen. 2009. *The Handbook of Communication Science*. Sage Publications.

[19] Stefanie Betz, Dominic Lammert, and Jari Porras. 2022. Software Engineers in Transition: Self-Role Attribution and Awareness for Sustainability.

[20] Stefanie Betz, Birgit Penzenstadler, Leticia Duboc, Ruzanna Chitchyan, Sedef Akinli Kocak, Ian Brooks, Shola Oyedeji, Jari Porras, Norbert Seyff, and Colin C. Venters. 2024. Lessons learned from developing a sustainability awareness framework for software engineering using design science. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–39.

[21] Seblewongel E. Biable, Nuno M. Garcia, and Dida Midekso. 2023. Proposed ethical framework for software requirements engineering. *IET Software* 17, 4 (2023), 526–537.

[22] Barry Boehm, Daniel Port, and Gail Kaiser. 2000. A combined curriculum research and curriculum development approach to software engineering education. In *Proceedings of the Conference on Software Engineering Education and Training*. IEEE Computer Society, 310–310.

[23] Alan Bond, Angus Morrison-Saunders, and Jenny Pope. 2012. Sustainability assessment: The state of the art. *Impact Assessment and Project Appraisal* 30, 1 (2012), 53–62.

[24] Markus Borg and Daniel Graziotin. 2024. Requirements for organizational resilience: Engineering developer happiness. *IEEE Software* 41, 4 (2024), 14–18.

[25] Klaus Bothe, Zoran Budimac, Rebeca Cortazar, Mirjana Ivanović, and Hussein Zedan. 2009. Development of a modern curriculum in software engineering at master level across countries. *Computer Science and Information Systems* 6, 1 (2009), 1–21.

[26] Pierre Bourque and Richard E. Fairley (Eds.). 2014. *SWEBOK: Guide to the Software Engineering Body of Knowledge* (version 3.0 ed.). IEEE Computer Society, Los Alamitos, CA. Retrieved from http://www.swebok.org/

[27] Katja Brundiers, Matthias Barth, Gisela Cebrián, Matthew Cohen, Liliana Diaz, Sonya Doucette-Remington, Weston Dripps, Geoffrey Habron, Niki Harré, Meghann Jarchow, et al. 2021. Key competencies in sustainability in higher education—Toward an agreed-upon reference framework. *Sustainability Science* 16 (2021), 13–29.

[28] AW Burton. 1998. *Movement Skill Assessment*. Human Kinetics (1998).

[29] José Jairo Camacho, J. M. Sanches-Torres, and Ernesto Galvis-Lista. 2013. Understanding the process of knowledge transfer in software engineering: A systematic literature review. In *The International Journal of Soft Computing and Software Engineering [JSCSE]. Special Issue: The Proceeding of International Conference on Soft Computing and Software Engineering 2013 (SCSE'13)*, Vol. 7545, 219–229. DOI: https://doi.org/10.7321/jscse.v3.n3.33

[30] Esther Cameron and Mike Green. 2019. *Making Sense of Change Management: A Complete Guide to the Models, Tools and Techniques of Organizational Change*. Kogan Page Publishers.

[31] Peter Checkland and John Poulter. 2020. Soft systems methodology. In *Systems Approaches to Making Change: A Practical Guide*, 201–253.

[32] Mykyta Chernenko. 2023. Lifelong learning. In *The Rational Software Engineer: Strategies for a Fulfilling Career in Tech*. Springer, 119–124.

[33] Yonghee Cho and Tugrul Daim. 2013. Technology forecasting methods. In *Research and Technology Management in the Electricity Industry: Methods, Tools and Case Studies*. Springer, 67–112.

[34] Mary Beth Chrissis, Mike Konrad, and Sandra Shrum. 2011. *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Pearson Education.

[35] David F. Ciambrone. 2018. *Environmental Life Cycle Analysis*. CRC Press.

[36] Kathryn R. Cieslak, Renee J. Anderson, Bart G. Hanson, Susan L. Korkowski, Debra K. Ness, and Jon T. Nordrum. 2011. A professional development track program: A case report. *HPA Resource* 11, 2 (2011).

[37] Andrea Crawford, Edward Sellman, and Stephen Joseph. 2021. Journaling: A more mindful approach to researching a mindfulness-based intervention in a junior school. *International Journal of Qualitative Methods* 20 (2021), 16094069211014771.

[38] John Cresswell. 2013. *Qualitative Inquiry & Research Design: Choosing Among Five Approaches*. Sage.

[39] Shirley Cruz, Fabio QB Da Silva, and Luiz Fernando Capretz. 2015. Forty years of research on personality in software engineering: A mapping study. *Computers in Human Behavior* 46 (2015), 94–113.

[40] Yousef-Awwad Daraghmi and Eman-Yasser Daraghmi. 2022. RAPD: Rapid and participatory application development of usable systems during COVID19 crisis. *IEEE Access* 10 (2022), 93601–93614. DOI: https://doi.org/10.1109/ACCESS.2022.3203582

[41] Pedro Domingos. 2015. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books.

[42] Harry Donkers. 2016. Integral Dynamics, A new integration of Wilber's integral theory and spiral dynamics. *International Journal of Humanities and Social Science* 6, 9 (2016), 6.

[43] Boru Douthwaite, Thomas Kuby, Elske Van De Fliert, and Steffen Schulz. 2003. Impact pathway evaluation: An approach for achieving and attributing impact in complex systems. *Agricultural Systems* 78, 2 (2003), 243–265.

[44] Victor Dulewicz and Malcolm Higgs. 2000. Emotional intelligence–A review and evaluation study. *Journal of Managerial Psychology* 15, 4 (2000), 341–372.

[45] Tobias Eisenreich, Sandro Speth, and Stefan Wagner. 2024. From requirements to architecture: An AI-based journey to semi-automatically generate software architectures. In *Proceedings of the 1st International Workshop on Designing Software*, 52–55.

[46] Clare Farrukh and Maria Holgado. 2020. Integrating sustainable value thinking into technology forecasting: A configurable toolset for early stage technology assessment. *Technological Forecasting and Social Change* 158 (2020), 120171.

[47] Robert Feldt, Francisco G. de Oliveira Neto, and Richard Torkar. 2018. Ways of applying artificial intelligence in software engineering. In *Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, 35–41.

[48] Maria Angela Ferrario and Emily Winter. 2023. Applying human values theory to software engineering practice: Lessons and implications. *IEEE Transactions on Software Engineering* 49, 3 (2023), 973–990. DOI: https://doi.org/10.1109/TSE.2022.3170087

[49] Mary Forehand et al. 2005. Bloom's taxonomy: Original and revised. *Emerging Perspectives on Learning, Teaching, and Technology* 8 (2005), 41–44.

[50] Batya Friedman, Peter Kahn, and Alan Borning. 2002. Value sensitive design: Theory and methods. *University of Washington Technical Report* 2, 8 (2002), 1–8.

[51] Vahid Garousi, Gorkem Giray, and Eray Tuzun. 2019. Understanding the knowledge gaps of software engineers: An empirical analysis based on SWEBOK. *ACM Transactions on Computing Education (TOCE)* 20, 1 (2019), 1–33.

[52] Gonzalo Génova, M. Rosario González, and Anabel Fraga. 2006. Ethical responsibility of the software engineer. In *PhiSE*.

[53] Gonzalo Génova, M. Rosario González, and Anabel Fraga. 2007. Ethical education in software engineering: Responsibility in the production of complex systems. *Science and Engineering Ethics* 13, 4 (2007), 505–522.

[54] Graham Gibbs. 1988. Learning by doing: A guide to teaching and learning methods. *Further Education Unit* (1988).

[55] Erving Goffman. 2023. The presentation of self in everyday life. In *Social Theory Re-wired*. Routledge, 450–459.

[56] Donald Gotterbarn. 1995. The moral responsibility of software developers: Three levels of professional software engineering. *Journal of Information Ethics* 4, 1 (1995), 54.

[57] D. W. Gotterbarn, Bo Brinkman, Catherine Flick, Michael S. Kirkpatrick, Keith Miller, Kate Vazansky, and Marty J. Wolf. 2018. ACM code of ethics and professional conduct.
[58] Don Gotterbarn and David Kreps. 2021. Being a data professional: Give voice to value in a data driven society. *AI and Ethics* 1, 2 (2021), 195–203.
[59] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2018. What happens when software developers are (un) happy. *Journal of Systems and Software* 140 (2018), 32–47.
[60] Lucas Gren and Robert Feldt. 2024. The social psychology of software security (Psycurity). arXiv:2407.00323. Retrieved from https://arxiv.org/abs/2407.00323
[61] Alicia M. Grubb. 2021. Four opportunities for SE ethics education. In *2021 IEEE/ACM 2nd International Workshop on Ethics in Software Engineering Research and Practice (SEthics)*. IEEE, 26–30.
[62] Armin Grunwald. 2009. Technology assessment: Concepts and methods. In *Philosophy of Technology and Engineering Sciences*. Elsevier, 1103–1146.
[63] Sivana Hamer, Christian Quesada-López, and Marcelo Jenkins. 2023. How have we researched developers' contributions in software engineering? A systematic mapping study. *SSRN* (2023).
[64] Jonathan M. Harris and Brian Roach. 2021. The theory of environmental externalities. In *Environmental and Natural Resource Economics*. Routledge, 44–92.
[65] Rogardt Heldal, Ngoc-Thanh Nguyen, Ana Moreira, Patricia Lago, Leticia Duboc, Stefanie Betz, Vlad C. Coroamă, Birgit Penzenstadler, Jari Porras, Rafael Capilla, et al. 2024. Sustainability competencies and skills in software engineering: An industry perspective. *Journal of Systems and Software* 211 (2024), 111978.
[66] April Hoang, Stevie-Jae Hepburn, Saya Tomizawa, Annemaree Carroll, Elizabeth Edwards, and Matthew Sanders. 2024. Using a stakeholder engagement approach to inform professional development programs to promote education for sustainability in schools. *Environmental Education Research* (2024), 1–19.
[67] Waqar Hussain, Harsha Perera, Jon Whittle, Arif Nurwidyantoro, Rashina Hoda, Rifat Ara Shams, and Gillian Oliver. 2022. Human values in software engineering: Contrasting case studies of practice. *IEEE Transactions on Software Engineering* 48, 5 (2022), 1818–1833. DOI: https://doi.org/10.1109/TSE.2020.3038802
[68] Jasmin Jahić and Ashkan Sami. 2024. State of practice: LLMs in software engineering and software architecture. In *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 311–318.
[69] Hannu-Matti Järvinen. 2017. Ethics as a skill of software engineer?
[70] Mehdi Jazayeri. 2004. The education of a software engineer. In *Proceedings of the 19th International Conference on Automated Software Engineering*. IEEE, xviii–xxvii.
[71] NorShahriza AbdulKarim, Fahda AlAmmar, and RomanaAziz. 2017. Ethical software: Integrating code of ethics into software development life cycle. In *2017 International Conference on Computer and Applications (ICCA)*. IEEE, 290–298.
[72] MarkKasunic. 2005. Designing an effective survey.
[73] RobertKegan. 1982. *The Evolving Self: Problem and Process in Human Development*. Harvard University Press.
[74] Asad Khan, Mohamad Noorman Masrek, and Fuziah Mohamad Nadzar. 2015. Analysis of competencies, job satisfaction and organizational commitment as indicators of job performance: A conceptual framework. *Education for Information* 31, 3 (2015), 125–141.
[75] Abby C. King, Diane K. King, Ann Banchoff, Smadar Solomonov, Ofir Ben Natan, Jenna Hua, Paul Gardiner, Lisa Goldman Rosas, Patricia Rodriguez Espinosa, Sandra J. Winter, et al. 2020. Employing participatory citizen science methods to promote age-friendly environments worldwide. *International Journal of Environmental Research and Public Health* 17, 5 (2020), 1541.
[76] David A Kolb. 2007. *The Kolb Learning Style Inventory*. Hay Resources Direct, Boston, MA.
[77] Matthew A. Koschmann and Jared Kopczynski. 2017. Stakeholder communication. In *The International Encyclopedia of Organizational Communication*, 1–13.
[78] Hannah Kosow and Robert Gaßner. 2008. *Methods of Future and Scenario Analysis: Overview, Assessment, and Selection Criteria*, Vol. 39. DEU.
[79] Markus Kröger. 2021. *Studying Complex Interactions and Outcomes Through Qualitative Comparative Analysis: A Practical Guide to Comparative Case Studies and Ethnographic Data Analysis*. Taylor & Francis.
[80] Amos Lal, Johnny Dang, Christoph Nabzdyk, Ognjen Gajic, and Vitaly Herasevich. 2022. Regulatory oversight and ethical concerns surrounding software as medical device (SaMD) and digital twin technology in healthcare. *Annals of Translational Medicine* 10, 18 (2022).
[81] G Le Ray and Pierre Pinson. 2020. The ethical smart grid: Enabling a fruitful and long-lasting relationship between utilities and customers. *Energy Policy* 140 (2020), 111258.
[82] Per Lenberg, Robert Feldt, and Lars-Göran Wallgren. 2014. Towards a behavioral software engineering. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, 48–55.

[83] Meira Levy and Irit Hadar. 2024. Requirements engineering for no-code development (RE4NCD) - A case study of rapid application development during War. In *Proceedings of the 30th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ '24)*. Daniel Méndez and Ana Moreira (Eds.), Lecture Notes in Computer Science, Vol. 14588, Springer, 322–329. DOI: https://doi.org/10.1007/978-3-031-57327-9_20

[84] Paul Luo Li, Amy J. Ko, and Jiamin Zhu. 2015. What makes a great software engineer? In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 700–710.

[85] Adam Lindgreen, C. Anthony Di Benedetto, Roderick J. Brodie, and Michel Van Der Borgh. 2020. How to undertake great cross-disciplinary research, 5 pages.

[86] Kaihua Liu, Sandeep Reddivari, and Kalyan Reddivari. 2022. Artificial intelligence in software requirements engineering: State-of-the-art. In *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 106–111.

[87] Yotam Lurie and Shlomo Mark. 2016. Professional ethics of software engineers: An ethical framework. *Science and Engineering Ethics* 22 (2016), 417–434.

[88] Alessia Maccaro, Katy Stokes, Laura Statham, Lucas He, Arthur Williams, Leandro Pecchia, and Davide Piaggio. 2024. Clearing the fog: A scoping literature review on the ethical issues surrounding artificial intelligence-based medical devices. *Journal of Personalized Medicine* 14, 5 (April 2024), 443. DOI: https://doi.org/10.3390/jpm14050443

[89] Katie MacLure, Andrew MacLure, Sharon Levy, and Wendy Dearing. 2022. Characterising the health and social care segment of the BCS (The Chartered Institute for IT) membership and their continuing professional development needs. *BMJ Health & Care Informatics* 29, 1 (2022).

[90] Murray Macza. 2008. A Canadian perspective of the history of process safety management legislation. In *8th Internationale Symposium Programmable Electronic System in Safety-Related Applications*, Vol. 22.

[91] Steven M. Manson, Shipeng Sun, and Dudley Bonsal. 2012. Agent-based modeling and complexity. In *Agent-Based Models of Geographical Systems*, 125–139.

[92] Senyeki Marebane, Robert Hans, and Potego Kgaphola. 2023. Mapping of studies on ethical responsibilities of software engineers. In *2023 IEEE AFRICON*. IEEE, 1–6.

[93] Gord McCalla. 2002. Software engineering requires individual professionalism. *Communications of the ACM* 45, 11 (2002), 98–101.

[94] Reuben R. McDaniel and Dean J. Driebe. 2001. Complexity science and health care management. In *Advances in Health Care Management*. Emerald Group Publishing Limited, 11–36.

[95] Stuart McDonald, Dave Towey, and Vladimir Brusic. 2022. Social impact of smart environments: Software engineering perspectives and challenges. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 1592–1597.

[96] Andrew McNamara, Justin Smith, and Emerson Murphy-Hill. 2018. Does ACM's code of ethics change ethical decision making in software development? In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 729–733.

[97] Donella Meadows. 2015. Leverage points-places to intervene in a system.

[98] Ali Mehraj, Zheying Zhang, and Kari Systä. 2024. A tertiary study on AI for requirements engineering. In *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 159–177.

[99] Harald A. Mieg. 2001. *The Social Psychology of Expertise: Case Studies in Research, Professional Domains, and Expert Roles*. Psychology Press.

[100] Alok Mishra and Deepti Mishra. 2021. Sustainable software engineering: Curriculum development based on ACM/IEEE guidelines. In *Software Sustainability*, 269–285.

[101] Anna Mitchell, Dharini Balasubramaniam, and Jade Fletcher. 2022. Incorporating Ethics in Software Engineering: Challenges and Opportunities. In *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 90–98.

[102] The Editors of Encyclopaedia. n.d. Encyclopaedia Britannica Role. Retrieved from https://www.britannica.com/topic/role

[103] Sadeeb S. Ottenburger, Rob Cox, Badrul H. Chowdhury, Dmytro Trybushnyi, Ehmedi Al Omar, Sujay A. Kaloti, Ulrich Ufer, Witold-R. Poganietz, Weijia Liu, Evgenia Deines, et al. 2024. Sustainable urban transformations based on integrated microgrid designs. *Nature Sustainability* (2024), 1–13.

[104] B Page. 1996. Environmental informatics towards a new discipline in applied computer science for environmental protection and research. In *Environmental Software Systems: Proceedings of the International Symposium on Environmental Software Systems*. Springer, 3–22.

[105] Aastha Pant, Rashina Hoda, Chakkrit Tantithamthavorn, and Burak Turhan. 2024. Ethics in AI through the practitioner's view: A grounded theory literature review. *Empirical Software Engineering* 29, 3 (2024), 67.

[106] David Lorge Parnas. 2002. Licensing software engineers in Canada. *Communications of the ACM* 45, 11 (2002), 96–98.

[107] Douglas Paton. 2003. Disaster preparedness: A social-cognitive perspective. *Disaster Prevention and Management: An International Journal* 12, 3 (2003), 210–216.

[108] Birgit Penzenstadler. 2020. Leverage points for focus flow and communitas. In *Proceedings of the 7th International Conference on ICT for Sustainability*, 267–274.

[109] Birgit Penzenstadler. 2020. Where attention goes, energy flows: Enhancing individual sustainability in software engineering. In *Proceedings of the 7th International Conference on ICT for Sustainability*, 139–146.

[110] Birgit Penzenstadler, Leticia Duboc, Colin C. Venters, Stefanie Betz, Norbert Seyff, Krzsztof Wnuk, Ruzanna Chitchyan, Steve M. Easterbrook, and Christoph Becker. 2018. Software engineering for sustainability: Find the leverage points! *IEEE Software* 35, 4 (2018), 22–33.

[111] Birgit Penzenstadler, Daniel Méndez Fernández, Debra Richardson, David Callele, and Krzysztof Wnuk. 2013. The requirements engineering body of knowledge (REBoK). In *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE, 377–379.

[112] Birgit Penzenstadler, Richard Torkar, and Cristina Martinez Montes. 2022. Take a deep breath: Benefits of neuroplasticity practices for software developers and computer workers in a family of experiments. *Empirical Software Engineering* 27, 4 (2022), 98.

[113] Anne-Kathrin Peters, Rafael Capilla, Vlad Constantin Coroamă, Rogardt Heldal, Patricia Lago, Ola Leifler, Ana Moreira, João Paulo Fernandes, Birgit Penzenstadler, Jari Porras, et al. 2023. Sustainability in computing education: A systematic literature review. *ACM Transactions on Computing Education* 24, 1 (2023), 1–53.

[114] Tina L. Peterson, Rodrigo Ferreira, and Moshe Y. Vardi. 2023. Abstracted power and responsibility in computer science ethics education. *IEEE Transactions on Technology and Society* 4, 1 (2023), 96–102.

[115] Klaus Pohl. 2016. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam-Foundation Level-IREB Compliant*. Rocky Nook, Inc.

[116] Valesca P. Retèl, Manuela A. Joore, Sabine C. Linn, Emiel J. T. Rutgers, and Wim H. van Harten. 2012. Scenario drafting to anticipate future developments in technology assessment. *BMC Research Notes* 5 (2012), 1–12.

[117] Lucy Danielle Robinson, Jade Lauren Cawthray, Sarah Elizabeth West, Aletta Bonn, and Janice Ansine. 2018. Ten principles of citizen science. In *Citizen science: Innovation in open science, society and policy*. UCL Press, 27–40.

[118] Guillermo Rodriguez-Navas, Leticia Duboc, Stefanie Betz, Ruzanna Chitchyan, Birgit Penzenstadler, and Colin Venters. 2015. Safety vs. sustainability design: Analogies, differences and potential synergies.

[119] Ina Schieferdecker. 2020. Responsible software engineering. In *The Future of Software Quality Assurance*, 137–146.

[120] Christoph Schneider and Stefanie Betz. 2022. Transformation^2: Making software engineering accountable for sustainability. *Journal of Responsible Technology* 10 (2022), 100027.

[121] Beth-Anne Schuelke-Leech and Marta Janczarski. 2019. Incorporating societal (social) and ethical implications into the design, development, and deployment of technologies. In *2019 IEEE International Symposium on Technology and Society (ISTAS)*, 1–6. DOI: https://doi.org/10.1109/ISTAS48451.2019.8937964

[122] Douglas Schuler and Aki Namioka. 1993. *Participatory Design: Principles and Practices*. CRC Press.

[123] Shalom H. Schwartz and Wolfgang Bilsky. 1987. Toward a universal psychological structure of human values. *Journal of Personality and Social Psychology* 53, 3 (1987), 550.

[124] Alexander Serebrenik, Kelly Blincoe, Byron Williams, and Joanne Atlee. 2020. Diversity and inclusion in the software engineering research community. *ACM SIGSOFT Software Engineering Notes* 45, 4 (2020), 5–7.

[125] Helen Sharp, Yvonne Dittrich, and Cleidson R. B. De Souza. 2016. The role of ethnographic studies in empirical software engineering. *IEEE Transactions on Software Engineering* 42, 8 (2016), 786–804.

[126] Mary Shaw. 2000. Software engineering education: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, 371–380.

[127] Thomas B. Sheridan. 1980. Computer control and human alienation. *Technology Review* 83, 1 (1980), 60–73.

[128] Erica S. Simmons and Nicholas Rush Smith. 2019. The case for comparative ethnography. *Comparative Politics* 51, 3 (2019), 341–359.

[129] Ian Sommerville. 2007. Sommerville: Software engineering.

[130] Kate Sparks, Brian Faragher, and Cary L. Cooper. 2001. Well-being and occupational health in the 21st century workplace. *Journal of Occupational and Organizational Psychology* 74, 4 (2001), 489–509.

[131] Klaas-Jan Stol and Brian Fitzgerald. 2018. The ABC of software engineering research. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 27, 3 (2018), 1–51.

[132] Margaret-Anne Storey, Neil A Ernst, Courtney Williams, and Eirini Kalliamvakou. 2020. The who, what, how of software engineering research: A socio-technical framework. *Empirical Software Engineering* 25 (2020), 4097–4129.

[133] Richard T. Stuebi. 2022. Ethical issues associated with multi-customer microgrids. *Research Handbook on Energy, Law and Ethics* (2022), 308–318.

[134] Anne Suryani. 2008. Comparing case study and ethnography as qualitative research approaches. *Jurnal Ilmu Komunikasi* 5, 1 (2008).

[135] Thien A. Tran and Tugrul Daim. 2008. A taxonomic review of methods and tools applied in technology assessment. *Technological Forecasting and Social Change* 75, 9 (2008), 1396–1405.
[136] William Truscott. 2012. *Six Sigma*. Routledge.
[137] Heiko A Von Der Gracht. 2012. Consensus measurement in Delphi studies: Review and implications for future quality assurance. *Technological Forecasting and Social Change* 79, 8 (2012), 1525–1536.
[138] Roel J. Wieringa. 2014. *Design Science Methodology for Information Systems and Software Engineering*. Springer.
[139] Ken Wilber. 2000. *Integral Psychology: Consciousness, Spirit, Psychology, Therapy*. Shambhala Publications.
[140] Robert K. Yin. 2009. *Case Study Research: Design and Methods*, Vol. 5. Sage.
[141] He Zhang, Xin Huang, Xin Zhou, Huang Grach, and Muhammad Ali Babar. 2019. Ethnographic research in software engineering: A critical review and checklist. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 659–670.
[142] H. Zhu, M. Zhou, and P. Seguin. 2006. Supporting software development with roles. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 36, 6 (2006), 1110–1123. DOI: https://doi.org/10.1109/TSMCA.2006.883170

Received 1 October 2024; revised 15 December 2024; accepted 18 December 2024