Nazar Kuchma
Kyiv National University of Technologies and Design
(Kyiv, Ukraine)
Maryna Vyshnevska
Kyiv National University of Technologies and Design
(Kyiv, Ukraine)

SOFTWARE ENGINEERING: FORMATION AND DEVELOPMENT

Since 2012, among the priority areas of education and science for the training of undergraduate and postgraduate students, internships of scientific and scientific-pedagogical workers in leading higher education institutions and scientific institutions abroad, related to informatics and computing, three - software engineering, systems software and software engineering - belong to one specialty: 121 - Software Engineering. In addition, a significant part of other priority areas (mathematical and computer modeling; information and communication technologies; artificial intelligence systems; systems programming, etc.) are related to it.

The term "software engineering" was deliberately chosen as provocative: "this concept meant that the production of software should be based on the same type of theoretical foundations and practical applications as in traditional branches of engineering." (Striuk, 2018, 12-13).

The software engineering profession stands at a critical juncture as artificial intelligence (AI) tools become increasingly capable of performing tasks traditionally reserved for human programmers. GitHub Copilot, which entered technical preview on June 29, 2021, and reached general availability on June 21, 2022, demonstrated that AI could assist with code generation at scale. Since then, numerous AI-powered development tools have emerged, raising fundamental questions about the future role of human software engineers.

This transformation differs qualitatively from previous technological advances in software development. While earlier innovations—from high-level programming languages to integrated development environments—enhanced human capabilities, AI

tools can independently generate functional code, potentially displacing certain categories of programming work. This paper proposes a theoretical framework for understanding and navigating this professional transformation. We argue that software engineers must evolve beyond traditional "code-first" approaches toward integrated business and technical competencies to maintain professional relevance and value (Mazumder Alauddin, 2025, 2).

The first use of the term “software engineering” dates back to August 1966, but the year of the emergence of software engineering (SEE) as a field is considered to be 1968, when the first conference called “Software engineering” was held in Germany. The main topic of the conference was the design, production and maintenance of software. One of the main participants of the conference, Peter Naur, noted that the work of software designers is similar to the work of architects and civil engineers, especially those who design large heterogeneous structures such as cities and industrial enterprises.

Competencies of software engineering specialists are formed through individualized learning, distance technologies, and electronic communication in the process of studying both general and specialized courses. This is due to the following needs: a high level of integration with the global educational system and labor market; the implementation of traditional and new forms of interaction between teacher and student in a virtual environment; the development of skills in applying modern IT to solve social and professional tasks.

To form a competitive specialist in software engineering, it is necessary to pay attention not only to the formation of knowledge in certain fundamental and professional disciplines, but also to organize the educational process in such a way as to maximally contribute to the development of certain personal qualities in students. Being a specialist is a process, not a phenomenon; you cannot become a highly qualified programmer at some point and never learn anything new. A specialist must have professional thinking, the main characteristics of which are a critical attitude to what has been achieved, the

ability to propose new things and the ability to take into account the influences of all significant internal and external factors.

Theoretical analysis made it possible to determine the functions, goals, and structure of the competence of future software engineering specialists as a component of their professional competence, which includes both activity-based and personal components (Hovorun, 2025, 34-135).

The evolution of software engineering practices has been both rapid and transformative, shaped by technological advancements and the increasing complexity of software demands. Since the early days of software development, methodologies have undergone continuous improvements, progressing from the traditional waterfall model to more iterative and agile practices. This evolution has aimed to address the growing need for efficiency, flexibility, and quality in software products. Over time, the adoption of DevOps, continuous integration, and test automation has redefined how software is built, tested, and deployed. Despite these advances, certain challenges such as error-prone manual coding, delayed feedback loops, and resource allocation have persisted, necessitating further innovation.

AI's transformative potential in software engineering is evident from tools like GitHub Copilot 1.7.4421. In a study, developers using Copilot completed coding tasks approximately 55% faster than those relying on traditional methods. Furthermore, the tool contributed to enhanced code quality, with higher rates of test case success on the first attempt. This underscores how AI-driven tools can significantly improve developer productivity while reducing errors in software development.

Ultimately, this research underscores that the integration of AI in software engineering is not merely a technical issue but also a strategic, organizational, and ethical one. By taking a holistic view, we can harness AI's transformative power responsibly, driving sustainable innovations in the field of software engineering.

The evolving relationship between AI and SE necessitates significant adjustments in educational curricula. Incorporating AI literacy into SE education ensures that upcoming professionals are equipped to leverage AI technologies effectively. AI virtual assistants and recommender systems can enhance learning experiences by providing personalized support and leveraging collective knowledge. By integrating AI-focused modules and practical applications into SE programs, educational institutions can prepare students to meet the demands of an AI-enhanced software development landscape.

Design patterns represent reusable solutions to recurrent problems in software design, and their appropriate application is instrumental in developing robust and maintainable systems. AI technologies can assist developers by automatically recognizing suitable design patterns based on system requirements and the existing codebase (Zhenyu Chen and Chunrong Fang, 2025,1-7).

REFERENCES

1. Hovorun V.V (2025). Analiz naukovykh dzherel shchodo formuvannia profesiinykh kompetentnostei u maibutnikh fakhivtsiv z inzhenerii prohramnoho zabezpechennia [Analysis of scientific sources on the formation of professional competences in future specialists in software engineering], pp.136-137[in Ukranian].
2. Mazumder, Md. Alauddin. (2025). The Evolution of Software Engineering Roles in the Age of AI: A Theoretical Framework for Professional Adaptation. 10.5281/zenodo.15577620.
3. Striuk, A.M. (2018). Inzheneriia prohramnoho zabezpechennia: pershi 50 rokiv stanovlennia ta rozvytku [Software engineering: the first 50 years of formation and development]. Computer Science & Software Engineering: Proceedings of the 1st Student Workshop (CS&SE@SW 2018), 2292, pp. 11–36 [in Ukrainian].
4. Zhenyu Chen and Chunrong Fang (2025), AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. pp. 1-7.