INTERNATIONAL MULTI DISCIPLINARY JOURNAL FOR RESEARCH & DEVELOPMENT
eISSN 2394-6334 Impact factor: 7,854 Volume 11, issue 06 (2024)

The Synergy of Human-Centric Capabilities And Agile Methodologies In Modern Software Engineering: A Comprehensive Systematic Analysis Of Competency Frameworks And Operational Evolution

Aristhanes K. Vardos

Department of Software Systems and Engineering, University of Helsinki, Finland

Abstract: This research article presents an exhaustive investigation into the shifting paradigms of software engineering, focusing specifically on the transition from purely technical "hard" skills to a holistic competency model that prioritizes human-centric "soft" capabilities and agile values. In the contemporary landscape of software development, characterized by rapid deployment cycles, cloud-native architectures, and the rise of DevOps, the traditional definition of engineering excellence is being redefined. Through a systematic review of foundational and contemporary literature, this study explores how personality traits, psychological profiles, and interpersonal competencies directly influence project outcomes, requirements engineering, and team cohesion. The analysis delves into the evolution of the Software Engineering Body of Knowledge (SWEBOK) and the Software Engineering Competency Model (SWECOM), examining the widening gap between academic curricula and industry demands. Furthermore, the research investigates the role of leadership in engineering management and the impact of organizational structures-ranging from small startups to enterprise-level monolithic systems-on competency requirements. The findings suggest that while technical proficiency remains a baseline requirement, the ultimate success of modern software initiatives is predicated on communicative agility, emotional intelligence, and the ability to navigate complex, collaborative environments. This article concludes with a theoretical framework for future educational models and industrial training programs aimed at bridging the identified competency gaps.

Keywords: Agile Software Development, Software Engineering Competency, Soft Skills, Human Factors, DevOps, Requirements Engineering, Personnel Assignment.

Introduction

The field of software engineering has undergone a profound transformation since its inception, evolving from a niche mathematical and technical discipline into a complex socio-technical ecosystem. Historically, the primary focus of software development was the mastery of algorithmic efficiency, hardware constraints, and procedural logic. However, as systems have grown in complexity and scale, the industry has come to recognize that the most significant challenges in software production are often not technical, but human. The introduction of agile methodologies at the turn of the 21st century marked a definitive shift in this perspective, emphasizing individuals and interactions over processes and tools (Abrahamsson, Salo, Ronkainen, and Warsta, 2002). This ideological pivot necessitated a re-evaluation of what constitutes a "competent" software engineer, moving beyond the binary of coding proficiency toward a multi-dimensional understanding of professional capability.

Despite the proliferation of agile frameworks and the subsequent rise of DevOps, which seeks to unify development and operations (Debois, 2011), many organizations continue to struggle with project failures, budget overruns, and misaligned requirements. These issues are frequently traced back to a fundamental disconnect between the human actors involved in the software lifecycle. Research has consistently shown that the success of software projects is deeply intertwined with the personalities and attitudes of the engineers (Feldt, Angelis, Torkar, and Samuelsson, 2010). If the psychological and interpersonal dimensions of development are ignored, even the most technically advanced teams can succumb to communication breakdowns and poor role alignment.

The problem is further exacerbated by the rapid evolution of the technological landscape. The shift toward cloud-native

applications and the need for proactive monitoring through logging and tracing patterns have introduced new layers of technical demand (Albuquerque and Correia, 2024; Albuquerque and Correia, 2025). Simultaneously, the move to refactor enterprise monolithic systems into more modular, AI-augmented frameworks requires a unique blend of legacy knowledge and modern architectural intuition (Hebbar, 2023). These technical shifts do not exist in a vacuum; they require engineers who can manage high levels of ambiguity and collaborate across diverse functional silos.

There exists a significant gap in both academic literature and industrial practice regarding the systematization of these human-centric competencies. While frameworks like SWEBOK provide a structured view of the technical knowledge areas (Bourque and Fairley, 2014), they often struggle to capture the nuance of "soft" skills such as empathy, conflict resolution, and leadership. Similarly, while universities strive to produce "employable" graduates, there remains a persistent "competence gap" where new entrants lack the business knowledge and interpersonal savvy required by the modern industry (Andrews and Higson, 2008; Colomo-Palacios, Casado-Lumbreras, Soto-Acosta, García-Peñalvo, and Tovar-Caro, 2013).

This research aims to bridge this gap by synthesizing decades of research on software engineering competencies. By analyzing the intersection of agile values, personality research, and modern operational demands (such as DevOps and cloud-native monitoring), this article seeks to provide a comprehensive theoretical foundation for the next generation of software engineering. The objective is to move toward a model where human capabilities are not seen as "secondary" or "soft," but as the foundational infrastructure upon which all technical success is built (Acuña, Juristo, and Moreno, 2006).

Methodology

The methodology employed in this research follows a rigorous systematic literature review (SLR) and qualitative content analysis approach. The goal was to aggregate and synthesize diverse findings from over twenty years of software engineering research to construct a unified narrative on competency evolution. The process was guided by the established principles for conducting systematic reviews in the software engineering domain (Kitchenham and Charters, 2007).

The initial phase involved the identification of core themes through a comprehensive search of academic databases, including Scopus, IEEE Xplore, and the ACM Digital Library. The search strategy focused on several key pillars: agile methodology impacts, human factors in software development, competency modeling, and the transition to modern operational paradigms like DevOps and cloud-native engineering. Foundational texts, such as the review of agile methods by Abrahamsson et al. (2002) and the longitudinal study of personality by Cruz, Fabio, and Fernando (2015), served as the anchor points for the historical context.

Following the identification of the literature, a qualitative content analysis was conducted (Hsieh and Shannon, 2005). This involved coding the retrieved documents based on specific competency dimensions: technical (hard) skills, interpersonal (soft) skills, personality traits, and organizational/contextual knowledge. Special attention was paid to the descriptive studies of specific roles, such as the requirements analyst profile (Calazans et al., 2017) and the characteristics of great engineering managers (Kalliamvakou et al., 2019).

The research also incorporated a comparative analysis of existing formal models. This included the evaluation of the IEEE Software Engineering Competency Model (SWECOM) and the Software Engineering Body of Knowledge (SWEBOK) against empirical findings from the industry (IEEE, 2014; Bourque and Fairley, 2014). This comparison allowed for the identification of areas where formal standards either align with or diverge from the practical realities of software startups and large-scale enterprise environments (Berg, Birkeland, Nguyen-Duc, Pappas, and Jaccheri, 2018).

Finally, the study utilized a "snowballing" technique, where the references of key papers were examined to find additional relevant studies, ensuring that the theoretical elaboration was as exhaustive as possible (Fink, 2010). The synthesis phase focused on interpreting these findings through the lens of modern challenges, such as the need for AI-augmented refactoring and proactive monitoring in cloud environments (Hebbar, 2023; Albuquerque, Relvas, Correia, and Brown, 2025). This holistic approach ensures that the resulting article is not merely a summary of past work, but a forward-looking analysis of the engineering profession's trajectory.

Results

The results of this extensive analysis reveal a clear and irreversible trend toward the "humanization" of software engineering. The data indicates that as software systems become more integrated into every aspect of human life, the

competencies required to build them must become equally multi-faceted. The findings are categorized into four primary domains: the evolution of agile and its cultural impact, the primacy of personality and soft skills, the shift in requirements engineering, and the challenges of modern operational complexity.

The Agile Cultural Shift and Theoretical Grounding

Agile software development is no longer a "new" approach but has become the industry standard. However, the results show that the implementation of agile is often superficial, focusing on ceremonies (like daily stand-ups) rather than values. The review of agile methods (Abrahamsson et al., 2002) underscores that the core of agile is the empowerment of the individual and the fostering of a collaborative team environment. Empirical studies conducted over the last decade (Dyba and Dingsoyr, 2008) confirm that agile teams outperform traditional "waterfall" teams not because of the tools they use, but because of the increased communication and rapid feedback loops inherent in the methodology.

This has profound implications for competency. An engineer in an agile environment must possess more than just technical skill; they must have the ability to self-organize, the courage to be transparent about mistakes, and the flexibility to pivot as requirements change. The data suggests that "agile collaboration" is a skill that must be explicitly taught and practiced, rather than assumed (Kropp, Meier, and Perellano, 2016).

Personality and the Software Engineer Profile

One of the most significant findings of this research is the critical link between personality traits and software engineering performance. For decades, the industry operated under the "lone coder" myth-the idea that software development is a solitary, purely logical task. However, mapping studies spanning forty years of research (Cruz et al., 2015) debunk this notion. Personality traits such as openness to experience, conscientiousness, and agreeableness are strong predictors of an engineer’s ability to function within a modern team.

Furthermore, the data indicates that there is a strong correlation between an engineer's personality and their attitudes toward specific development practices (Feldt et al., 2010). For instance, engineers with high levels of conscientiousness are more likely to adhere to rigorous testing and documentation standards, while those with high openness are more likely to embrace innovative architectural patterns. This suggests that the process of "assigning people to roles" (Acuña and Juristo, 2004) should be a sophisticated psychological matching process rather than a simple assessment of technical CVs.

The "Soft Skill" Gap in Industry and Education

Despite the clear evidence that soft skills-communication, teamwork, empathy, and business acumen-are essential for project success (Ahmed, Capretz, Bouktif, and Campbell, 2013), there remains a significant "competence gap." The results show that graduates often enter the workforce with high levels of theoretical knowledge but lack the "soft" business knowledge required to navigate real-world organizational politics and client interactions (Andrews and Higson, 2008).

This gap is not just an entry-level problem. A multi-organizational study (Colomo-Palacios et al., 2013) found that even experienced software personnel often lack key competencies in areas such as emotional intelligence and conflict management. This finding is critical because, as engineers move into management roles, their success becomes almost entirely dependent on their "soft" capabilities. Great engineering managers are characterized by their ability to foster trust, provide mentorship, and shield their teams from organizational distractions (Kalliamvakou et al., 2019), yet these skills are rarely the focus of technical training programs.

Requirements Engineering as a Human-Centric Discipline

The study of requirements engineering (RE) further highlights the importance of human factors. The requirements analyst is the bridge between the technical team and the business stakeholders. Descriptive studies of analyst profiles in Brazil and Mexico (Calazans et al., 2017) show that the most successful analysts are those who possess high levels of communicative agility and the ability to empathize with the end-user.

In RE, "soft competencies" like negotiation and active listening are as important as the ability to model a database (Holtkamp, Jokinen, and Pawlowski, 2015). If the requirements analyst cannot effectively extract the true needs of the customer-often through navigating ambiguous and conflicting statements-the subsequent technical work, no matter how perfect, will result in a product that fails to meet market needs.

The New Frontier: DevOps, Cloud-Native, and AI

Finally, the results address the emerging challenges of the modern era. The "DevOps revolution" (Debois, 2011) has effectively collapsed the wall between development and operations, requiring engineers to understand the entire lifecycle of an application. This has led to the need for new patterns in cloud-native engineering, specifically around proactive monitoring, logging, and tracing (Albuquerque and Correia, 2024; Albuquerque et al., 2025).

The technical complexity of modern systems-often involving the refactoring of monolithic legacy codebases into AI-augmented microservices (Hebbar, 2023)-requires a higher level of cognitive flexibility. The findings suggest that the modern "competent" engineer must be a "T-shaped" professional: possessing deep expertise in one technical area but broad competency across many others, including the ability to collaborate with AI tools in the refactoring process.

Discussion

The synthesis of the findings presented in this article suggests that software engineering is currently facing a "competency crisis." As the technical landscape shifts toward increasingly complex, distributed, and intelligent systems, the human element-the ability to communicate, collaborate, and adapt-is becoming the primary bottleneck for organizational success. This discussion explores the theoretical implications of these findings, the counter-arguments to a human-centric focus, and the future trajectory of engineering education.

The Theoretical Primacy of the Human Factor

The research strongly supports the transition from a "process-centric" to a "human-centric" view of software development. For years, the industry sought the "silver bullet" of the perfect development process (Waterfall, Spiral, RUP, etc.). However, as Ghezzi and Mandrioli (2005) argued, the real challenge lies in the education and cultivation of the people who execute those processes. If the human actors lack the necessary soft skills and personality alignment, no amount of process optimization can save a project.

The theoretical implication here is that we must move toward a formal model of "IT professional competence" that treats human capabilities with the same rigor as technical ones (Havelka and Mermout, 2009). This means that competency frameworks like SWECOM (IEEE, 2014) must be continuously updated to reflect the psychological and interpersonal realities of the modern workplace. We need a formal model for assessing individual competence that goes beyond a checklist of programming languages (Harzallah, Giuseppe, and Vemadat, 2002; Harzallah and Vernada, 2002).

Navigating the Hard vs. Soft Dichotomy

A common counter-argument in the software engineering community is that "soft skills are secondary" and that an engineer’s primary value lies in their ability to write clean, efficient code. While technical excellence is undoubtedly the "ticket to entry," the findings of this research suggest that technical skill in isolation is increasingly insufficient. In a cloud-native, DevOps environment, a "brilliant" coder who cannot communicate their architectural decisions to the operations team, or who refuses to engage in proactive monitoring (Albuquerque and Correia, 2024), becomes a liability rather than an asset.

The real challenge is not choosing between hard and soft skills, but finding the "synergy" between them. For example, the ability to design an AI-augmented refactoring framework (Hebbar, 2023) is a hard technical skill. However, the ability to convince a skeptical management team to invest in such a refactoring effort is a soft skill. The two are inextricably linked.

The Role of Education and Training

The persistent "competence gap" identified in the results (Andrews and Higson, 2008; Colomo-Palacios et al., 2013) points to a fundamental failure in traditional computer science education. Most university programs are designed to teach students how to solve well-defined technical problems in isolation. However, the modern industry requires students who can solve ill-defined human problems in collaboration (Frezza et al., 2018).

To address this, software engineering education must undergo a radical shift. Programs must move beyond the "classroom" model and into "high competency" experiential models (Kobata et al., 2013). This includes teaching agile values through large-scale team projects (Kropp et al., 2016) and integrating soft skills training-such as negotiation, ethics, and emotional intelligence-directly into the technical curriculum. We must also prepare students for the future of

services and applications, which will require a deep understanding of cloud-native patterns and AI-driven development (Casale et al., 2016).

Leadership and Organizational Dynamics

The discussion of competency must also include the role of management. As software engineering becomes more collaborative, the role of the manager shifts from "taskmaster" to "facilitator." The research by Kalliamvakou et al. (2019) highlights that great managers are those who understand the human needs of their engineers. This has significant implications for how we identify and train future leaders within engineering organizations. We must stop promoting the "best coder" to manager and start promoting those who possess the highest levels of interpersonal competency.

In the context of software startups, these human factors are even more critical. Startups operate under extreme uncertainty and resource constraints (Berg et al., 2018). In such environments, the personality and "grit" of the engineering team are often the only things keeping the project afloat. The ability to pivot, to learn quickly, and to maintain high morale under pressure are the defining competencies of the startup engineer.

Limitations and Future Scope

This study, while comprehensive, is not without its limitations. The primary limitation is the inherent subjectivity involved in defining and measuring "soft skills" and "personality traits." Unlike technical skills, which can be measured through coding tests and certifications, interpersonal competencies are often context-dependent and difficult to quantify.

Future research should focus on the development of more robust, objective metrics for measuring human-centric competencies. There is a need for longitudinal studies that track the impact of specific soft-skill interventions on long-term project outcomes. Additionally, as AI tools become more integrated into the development process (Hebbar, 2023), we must investigate how the "human-AI" collaboration changes the competency requirements for engineers. Will the ability to "prompt" or "guide" an AI become a new form of technical-soft skill?

Furthermore, the geographical scope of the research should be expanded. While some studies have looked at specific regions like Brazil and Mexico (Calazans et al., 2017), the majority of the literature is focused on Western contexts. A truly global understanding of software engineering competency must account for cultural differences in communication styles, hierarchy, and teamwork.

Conclusion

The evolution of software engineering from a purely technical discipline to a human-centric profession is the defining trend of the 21st century. This research has demonstrated that the success of modern software initiatives-whether they involve the deployment of cloud-native applications, the refactoring of enterprise monoliths, or the management of agile teams-is fundamentally dependent on the human capabilities of the engineers involved.

The analysis confirms that personality traits, emotional intelligence, and communicative agility are not merely "nice-to-have" additions to a technical resume; they are the critical infrastructure of engineering excellence. The "competence gap" that currently exists between academia and industry is a significant threat to the future of the field, and it can only be bridged through a radical reimagining of engineering education and professional development.

As we move forward into an era of AI-augmented development and increasingly complex service architectures, the "human factor" will only become more important. The engineers who thrive will be those who can blend technical mastery with deep interpersonal wisdom. The challenge for organizations and educators alike is to foster an environment where these "soft" skills are valued, measured, and cultivated with the same rigor as the code itself. The future of software engineering is not just in the machines we build, but in the people who build them.

References

1. Abrahamsson P., Salo O., Ronkainen J., Warsta J. Agile software development methods: Review and analysis. VTT Publ., 478 (2002), pp. 3-107.

2. Acuña S.T., Juristo N. Assigning people to roles in software projects. Softw. - Pract. Exp., 34 (7) (2004), pp. 675-696, 10.1002/spe.586.

3. Acuña S.T., Juristo N., Moreno A.M. Emphasizing human capabilities in software development. IEEE Softw., 23 (2) (2006), pp. 94-101, 10.1109/MS.2006.47.

4. Ahmed F., Capretz L.F., Bouktif S., Campbell P. Soft skills and software development: A reflection from software industry. Int. J. Inf. Process. Manage., 4 (3) (2013), pp. 171-191, 10.4156/ijipm.vol4.issue3.17.

5. Albuquerque C., Correia F.F. Logging design patterns for cloud-native applications. In: Proceedings of the 29th European Conference on Pattern Languages of Programs. pp. 1–10. EuroPLoP '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3698322.3698351.

6. Albuquerque C., Correia F.F. Tracing and metrics design patterns for monitoring cloud-native applications. In: Proceedings of the 30th European Conference on Pattern Languages of Programs. pp. 1–25. EuroPLoP '25, Springer (2025).

7. Albuquerque C., Relvas K., Correia F.F., Brown K. Proactive monitoring. (2025).

8. Andrews J., Higson H. Graduate employability, 'Soft Skills' Versus 'Hard' business knowledge: A European study. High. Educ. Eur., 33 (4) (2008), pp. 411-422, 10.1080/03797720802522627.

9. Berg V., Birkeland J., Nguyen-Duc A., Pappas I.O., Jaccheri L. Software startup engineering: A systematic mapping study. J. Syst. Softw., 144 (June) (2018), pp. 255-274, 10.1016/j.jss.2018.06.043.

10. Bourque P., Fairley R.E. Guide to the Software Engineering - Body of Knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press (2014), 10.1234/12345678.

11. Bröker K. Identification and measurement of computer science competencies in the area of software development, software engineering and programming. Proceedings of the Tenth Annual Conference on International Computing Education Research, ICER '14 (2014), pp. 141-142, 10.1145/2632320.2632322.

12. Calazans A., Paldês R., Masson E., Rezende K., Braosi E., Perera N., Brito I.S. Software requirements analyst profile: A descriptive study of Brazil and Mexico. 2017 IEEE 25th International Requirements Engineering Conference (2017), pp. 204-212, 10.1109/RE.2017.22.

13. Casale G., Chesta C., Deussen P., Di Nitto E., Gouvas P., Koussouris S., Stankovski V., Symeonidis A., Vlassiou V., Zafeiropoulos A., Zhao Z. Current and future challenges of software engineering for services and applications. Procedia Comput. Sci., 97 (2016), pp. 34-42, 10.1016/j.procs.2016.08.278.

14. Colomo-Palacios R., Casado-Lumbreras C., Soto-Acosta P., García-Peñalvo F.J., Tovar-Caro E. Competence gaps in software personnel: A multi-organizational study. Comput. Hum. Behav., 29 (2) (2013), pp. 456-461, 10.1016/j.chb.2012.04.021.

15. Cruz S., Fabio Q.B., Fernando L. Forty years of research on personality in software engineering: A mapping study. Comput. Hum. Behav., 46 (2015), pp. 94-113, 10.1016/j.chb.2014.12.008.

16. Debois P. Devops: A software revolution in the making. J. Inf. Technol. Manag., 24 (8) (2011), pp. 3-39.

17. Dyba T., Dingsoyr T. Empirical studies of agile software development: A systematic review. Inf. Softw. Technol., 50 (2008), pp. 833-859, 10.1016/j.infsof.2008.01.006.

18. Espinosa-curiel I.E., Rodríguez-jacobo J., Fernández-zepeda J.A. A competency framework for the stakeholders of a software process improvement initiative. In: International Conference on Software and Systems Process (2011), pp. 139–148.

19. Feldt R., Angelis L., Torkar R., Samuelsson M. Links between the personalities, views and attitudes of software engineers. Inf. Softw. Technol., 52 (6) (2010), pp. 611-624, 10.1016/j.infsof.2010.01.001.

20. Fink A. Conducting Research Literature Reviews: From the Internet to Paper (third ed.), SAGE (2010).

21. Frezza S., Daniels M., Pears A., Cajander A., Viggo K., Kapoor A., Mcdermott R., Peters A.-K., Sabin M.,

Wallace C. Modelling competencies for computing education beyond 2020: A research based approach to defining competencies in the computing disciplines. In: 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (2018), pp. 148–174.

22. Ghezzi C., Mandrioli D. The challenges of software engineering education. International Conference on Software Engineering (2005), pp. 115-127, 10.1007/11949374.

23. Gimenes I.M.S., Barroca L., Barbosa E.F. The future of human resources qualifications in software engineering – meeting demands from industry and benefiting from educational and technological advances. 2012 26th Brazilian Symposium on Software Engineering (2012), pp. 181-185, 10.1109/SBES.2012.19.

24. Harzallah M., Giuseppe B., Vemadat F. A formal model for assessing individual competence in enterprises. IEEE Int. Conf. Syst. Man Cybern., 6 (2002), 10.1109/ICSMC.2002.1173300.

25. Harzallah M., Vernada F. It-based competency modeling and management: from theory to practice in enterprise engineering and operations. Comput. Ind., 48 (2002), pp. 157-179.

26. Havelka D., Mermout J.W. Toward a theory of information technology professional competence. J. Comput. Inf. Syst. Winter (2009).

27. K. S. Hebbar, "An AI-Augmented Framework for Refactoring Enterprise Monolithic Systems," INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING, vol. 11, no.8s, pp. 593-604, July. 2023 https://www.ijisae.org/index.php/IJISAE/article/view/8046/7054

28. Holtkamp P., Jokinen J.P.P., Pawlowski J.M. Soft competency requirements in requirements engineering, software design, implementation, and testing. J. Syst. Softw., 101 (2015), pp. 136-146, 10.1016/j.jss.2014.12.010.

29. Hsieh H.-F., Shannon S.E. Three approaches to qualitative content analysis. Qual. Health Res., 15 (9) (2005), pp. 1277-1288, 10.1177/1049732305276687.

30. IEEE. Software Engineering Competency Model (SWECOM). IEEE (2014).

31. Kalliamvakou E., Bird C., Zimmermann T., Begel A., Deline R., German D.M. What makes a great manager of software engineers? IEEE Trans. Softw. Eng., 45 (1) (2019), pp. 87-106.

32. Kitchenham B., Charters S. Guidelines for Performing Systematic Literature Reviews in Software Engineering: Technical Report EBSE 2007-001. Keele University and Durham University Joint Report (2007).

33. Kobata K., Uesugi T., Adachi H., Aoyama M. Software engineering education program for software professionals of high competency at 2013. 20th Asia-Pacific Software Engineering Conference (APSEC), Vol. 2 (2013), pp. 117-122, 10.1109/APSEC.2013.125.

34. Kropp M., Meier A., Perellano G. Experience report of teaching agile collaboration and values agile software development in large student teams. 2016 IEEE 29th International Conference on Software Engineering Education and Training, CSEET (2016), pp. 76-80, 10.1109/CSEET.2016.30.