Contemporary Software Modernization: Perspectives and Challenges to Deal with Legacy Systems

Wesley Klewerton Guez Assunção
Johannes Kepler University Linz, Austria
Pontifical Catholic University of Rio de Janeiro, Brazil
wesley.assuncao@jku.at

***Abstract*—Software modernization is an inherent activity of software engineering, as inevitably systems become old and technology advances. The term "software modernization" emerged as a research topic in the early 2000's, with a differentiation of the traditional software evolution. Studies on this topic became popular due to new programming paradigms, technologies, and architectural styles. Due to the pervasive nature of software nowadays, modernizing legacy systems is paramount to provide users with competitive and innovative products and services. Despite the many pieces of work existing in literature, there are some limitations in this topic: (i) proposed approaches are too specific for one scenario or technology lacking flexibility, (ii) most of proposed approaches are not aligned with the current modern software development scenario, and (iii) based on a myriad of potential modernization approaches, practitioners may be misguided on how to modernize legacies. In this work, our goal is to call the attention to the need of advances in the research and practices towards well-defined software moderation field. The focus is to enable companies to preserved knowledge represented in legacy systems while employing disruptive and emerging technologies to benefit users. Based on that, we contextualize the different perspectives of software modernization in the context of contemporary software development. Also, we introduce a research agenda with 10 challenges to be taken into account.**

***Index Terms*—Legacy applications, software aging, reverse engineering, Re-Engineering, Software Migration.**

I. Introduction

Throughout the software life cycle, the architecture decay and degrade, the user requirements change, technologies evolve, and new business models emerge, leading to what are called legacy systems [1]. The large majority of existing industrial software are long-lived systems that represent several years of competitive knowledge and business value [2]. However, due to extensive maintenance and obsolete technology, legacy systems are costly to maintain, more exposed to cybersecurity risks, less effective in meeting their intended purpose, and push up costs of digital transformation [3], [4], [5]. For instance, the US government spent over $90 billion in fiscal year 2019 on IT, from which about 80% was used to operate and maintain legacy systems [3]. Also, the UK government spends £4.7 billion a year on IT across all departments, and £2.3 billion goes on patching up systems, some of which date back 30 years or more [4]. To remain competitive, companies must modernize their legacy systems, preserving the hard-earned knowledge acquired through many years of system development [2], [6], [7].

According to Seacord et al. [6] "*Software modernization attempts to evolve a legacy system, or elements of the system, when conventionally evolutionary practices, such maintenance and enhancement, can no longer achieve the desired system properties.*" The process of modernizing a legacy system leads to benefits such as easing engineering activities, satisfying user needs, achieving new business goals, or reducing costs [6]. Also, modernization is a mean to leverage the digital transformation [5], [8], as it allows using emerging/disruptive technologies such as artificial intelligence, high-performance computing, cloud computing, IoT, robotics, and big data.

In the literature, we can find several modernization strategies [8], [9]. For example, restructuring systems using components, adoption of aspect-oriented development, re-engineering of system variants into software product lines, migration to microservices, and supporting for new hardware, e.g., multi-core. Even the software development process has been modernized, e.g., DevOps. We can also observe that the modernization has different driving forces and impacts related to *organizational*, *operational*, and *technological* aspects. For instance, the modernization can focus on independence for agile teams, optimize the deployment, ease the inclusion of innovation, facilitate scalability, or explore new market segment [7], [9].

Despite the existing studies on the topic of modernization [7], [9], covering different strategies and aspects, there still are some limitations and gaps in the state of the art and practice: (i) proposed approaches are too specific for one scenario or technology, without flexibility, making hard their reusability or adaptation for different scenarios; (ii) most of proposed approaches are not aligned with the modern software development scenario, as there is no contemporaneous body of knowledge on the fundamentals of software modernization; and (iii) the existence of a myriad of different modernization strategies, on one hand, offers a wide range of potential solutions, however, on the other hand, such diversity of strategies may misguide practitioners, providers, and researchers when looking for solutions for specific situations.

The pieces of work that try to organize the existing studies on software modernization have several limitations. They only present an overview of the state of the art [8]; are based on few case studies or a subset of existing literature [9], [10], [11]; are outdated regarding current emerging/disruptive technologies [2], [11], [12], [13]; partially cover the modernization life cycle, and rarely take into account organizational, operational, and technological aspects [7], [8]. As pieces of work span across many years and focus on modernizing for different purposes, there is a need for a discussion on how these different modernization strategies can support the contemporaneous software development.

In this work, our goal is to call the attention to the need of advances in the research and practices towards software modernization in the light of contemporary software development. The focus is to enable companies to preserved knowledge represented in legacy systems while employing disruptive and emerging technologies to benefit users. Based on that, we contextualize the different perspectives of software modernization in the context of contemporary software development. Also, we introduce a research agenda with 10 challenges to be taken into account. The contribution is to motivate the discussion on how to perform software modernization, avoiding practitioners to adopt solutions because of popularity (hype) but that not fit correctly in their scenarios.

II. Background and Related Work

Seacord et al. [6] presented software modernization as a remedy to face the legacy system crisis in the early 2000's. They discussed how to keep or add business value through modern technologies, reducing operational costs, and dealing with technical aspects, as for example, allowing better reuse and easier maintenance [6]. However, their discussion is not totally aligned with current technological and operational advances of contemporary software engineering.

To decide for which modernization strategy to adopt, companies should perform a portfolio analysis. Figure 1 presents the portfolio analysis quadrant extended from Seacord et al. [6] to bring a contemporaneous perspective of software modernization. In addition to *technical quality* and *business value* dimensions, which range from low to high, we introduced *innovation* that is motivated by new disruptive and emerging technologies, which is a driving force for the modernization.

The five quadrants presented in Figure 1 are:

- **1 Replace**: legacy systems that have low business value and low technical quality, i.e., accumulated technical debt, should be replaced by new systems, using generic solutions or off-the-shelf systems, instead of undergo for a re-engineering or migration process.
- **2 Maintain**: systems with high technical quality and low business value should not require modernization effort, but go thought traditional maintenance activities, just to keep them operating and meeting customers need.
- **3 Evolve**: high-quality legacies with high business value should be actively evolved using traditional evolutionary development practices for introducing new features, new products, or even serving as third part for other systems.
- **4 Re-engineer**: systems with high business value and low technical quality should be re-engineered in order to preserve business values, i.e., external quality, and manage the technical debt, i.e., internal quality. This type of modernization can be transparent to the final user.

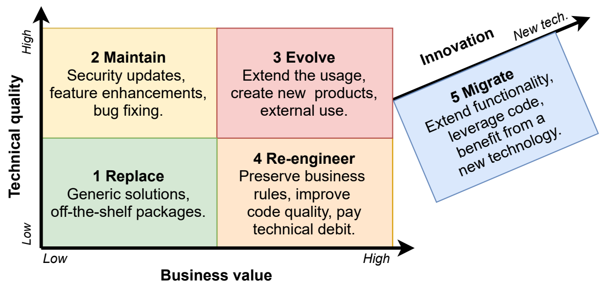
High
Technical quality
Low
2 Maintain
Security updates, feature enhancements, bug fixing.
3 Evolve
Extend the usage, create new products, external use.
1 Replace
Generic solutions, off-the-shelf packages.
4 Re-engineer
Preserve business rules, improve code quality, pay technical debit.
Low
High
Business value
Innovation
New tech.
5 Migrate
Extend functionality, leverage code, benefit from a new technology.

Fig. 1. Extended quadrant of the portfolio analysis for the contemporary software modernization, adapted from [14].

- ***5 Migrate***: when the system has high business value and a company decide to include innovation with emerging/disruptive technologies, independently of system's technical quality, the migration to the desired new technologies should take place. This is, for example, the case of companies that want to undertake digitalization.

In the literature, we can find *several modernization strategies* to retain business value of legacy systems [8], [9]. For example, restructuring systems using components [15], [16], [17], adoption of aspect-oriented development [18], [19], [20], re-engineering of system variants into software product lines [21], [22], [23], [24], migration to microservices [7], [25], [26], [27], [28], [29], supporting new devices or pieces of hardware, e.g., from single-core to multi-core machine [30], [31], [32], and classical information systems to quantum computing [33]. Even the software development process has been modernized, e.g., DevOps [34], [35]. Also, modernization has different driving forces and impacts related to *organizational*, *operational*, and *technological* aspects. For instance, the modernization can focus on, independence of teams, optimizing deployment, easing innovation, facilitating scalability, or exploring new market segment [7], [9].

Despite many studies, existing body of knowledge on software modernization have several limitations: they only present an overview of the state of the art [8]; are based on few case studies or a subset of existing literature [9], [10], [11]; are outdated regarding current emerging/disruptive technologies [2], [11], [12], [13]; partially cover the modernization life cycle, and rarely take into account *organizational*, *operational*, and *technological* aspects [7], [8]. Furthermore, these studies are limited on exploring contemporary needs, e.g., digital transformation [8]. Finally, software modernization must be seen as a multi-perspective activity, which is discussed next.

III. Multi-perspective and Challenges

Based on the state of the art and practice, and our previous work, we describe the multi-perspective of software modernization in the context of contemporary software development. Figure 2 presents six perspectives that affects the process of modernizing a legacy system. These perspectives range from understanding the legacy system, to conduct the transition from the legacy (or part of) to the modern system. Based on

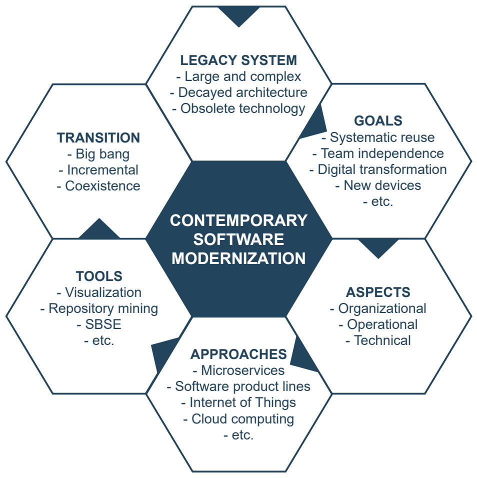

Fig. 2. Different perspectives of software modernization in the context of contemporary software development.

this multi-perspective, needs, trends, and recent pieces of work in the topic of software modernization, we present a list of challenges (C) to be taken into account in future work.

C_1: Lack of a comprehensive and contemporaneous body of knowledge on software modernization. The pieces of work that try to organize the existing body of knowledge on software modernization have several limitations, as discussed in Section II. Based on that, there is a need for a comprehensive and contemporaneous body of knowledge about these modernization strategies. We do not have to reinvent the wheel but organize existing knowledge on the light of the perspectives presented in Figure 2 and the contemporary software development.

C_2: Recommend the right approach based on the modernization goal. In Figure 2 we present examples of goals that can serve as driving forces for the modernization. Based on specific goals, some approaches could be more recommended than others. However, this recommendation must be an informed decision. The challenge here is to have guidelines to support practitioners and companies on how to choose the proper approach according to their goals, avoiding deciding only based on the technologies in the "hype". For example, microservices has been advertised as a solution for technology flexibility. However, a recent study has shown that this is not the most common driving forces to migrate to microservices [7]. Choosing the wrong approach can lead to frustrating modernization. For example, systems migrated to microservice going back to monolithic applications [36].

C_3: Establish hybrid environments to allow the legacy and modern parts of a system operating together. In Figure 2 we can see the three types of transition: (i) big bang, a.k.a. cold turkey [37], which is mainly the replacement of the legacy system with a modern one (see Figure 1); (ii) incremental modernization, following a strangler pattern [38], in which parts of the legacy systems are incremental replaced by modern parts [6], [7]; and (iii) the coexistence, in which the practitioners do not want to modernize all the legacy, then the legacy and modern parts still operate together in the system [39]. The big bang and incremental transitions are explored in literature, however, little is discussed on how to establish a hybrid environment to allow the development and coexistence of legacy and modern system.

C_4: Consider technical, operational, and organizational aspects during the modernization. The great majority of studies on software modernization discusses the technical aspects of the modernization [21], [26]. However, modernization has different driving forces and impacts related to *organizational*, *operational*, and *technological* aspects [6], [7], [9], [14]. Software engineering involves technologies, people, operations, and business strategy [40]. The challenge here is to propose approaches that deal with all these aspects.

C_5: Decide among replace, maintain, evolve, re-engineering, or migrate. As presented in Figure 1, there are different forms of modernization. Also, in Figure 2 we see that the legacy systems can present different problem related to its technical quality. Based on that, we can observe that choosing how to modernize a legacy system is a multi-criteria decision. Thus, companies need solutions to deal with this challenge.

C_6: Support digital transformation. Digital Transformation is currently a trend and have receiving great attention around the world. For example, the Europe Union has the Digital Europe Programme[1], Australia has the Digital Economy Strategy[2], in North America the Canada Digital Adoption Program[3] and the Digital Strategy[4] of the United States, and in Asia 11 countries have joined forces in the Connecting Capabilities[5]. Despite expected benefits, it is acknowledged that the digital transformation is hampered legacy systems, processes, and mindsets [5]. In this context, modernization is a mean to leverage the digital transformation [5], [8]. However, there are no guidelines on how to perform software modernization to leverage digital transformation. Yet, the exiting few pieces of work on this topic only provide a superficial overviews.

C_7: Prepare the legacy for the modernization. When the legacy system has a high business value, it is a candidate to modernization by re-engineering or migration, independently of its internal quality (see Figure 1). However, understanding and modernizing a legacy system with poor internal quality is a complex task. For example, systems usually evolve in space, adding new features, and time, with feature being revised [41], which make its comprehension difficult. For such a situation, we believe that using refactoring strategies can be a good way to improve the legacy internal quality to face the modernization. However, the literature is scarce on how this "pre-modernization" activity should take place.

C_8: Propose non-intrusive approaches and tools. Practitioners usually have preferences for using some technologies,

[1]https://digital-strategy.ec.europa.eu/en/activities/digital-programme
[2]https://digitaleconomy.pmc.gov.au/
[3]https://www.ic.gc.ca/eic/site/152.nsf/eng/home
[4]https://www.state.gov/digital-government-strategy/
[5]http://connectedfuture.economist.com/connecting-capabilities/

tools, and workflows. Based on that, researchers should propose modernization approaches and tools that take into account these preferences. In other words, as much as new approaches and tools are non-intrusive to already adopted development preferences, easier to transfer them to practice [42].

C_9: Train workforce with skills for dealing with modernization. Figure 2 presented the different perspectives of the software modernization. These several perspectives must be considered to educate practitioners to operationalize the modernization process [43]. Based on that, a challenge is to training the workforce with expertise to deal with the complexity of software modernization [33].

C_{10}: Modernization for small and medium-sized enterprises (SMEs). In the literature we observe that some software engineering activities should be conducted differently in the context of SMEs [44], [45]. This might also be the case for software modernization [46]. Based on that, research need to be conducted to deal with challenges faced by SMEs when modernizing their legacy systems in order to grow and be more competitive.

IV. Conclusion

Software modernization is a fundamental activity of software engineering, since inevitably requirements change, and technology advances, and new business models emerge. Despite that, research on this topic has not been following the modern software development, and legacy systems still remain a problem. To fill this gap and to sparkle the discussion on this topic, we present a discussion of software modernization in the light of contemporary software modernization. We revisited some pieces of work and introduce the multi-perspective of contemporary software modernization. Based on that, in this work we discussed 10 challenges to motivate and guide to new studies.

Disclaimer

This is a **non peer-reviewed paper**. All text and ideas discussed in this document are responsibility of the author. To cite this paper, please use this bibtex: https://wesleyklewerton.github.io/publications/SoftwareModernization.bib

References

[1] K. Bennett, "Legacy systems: coping with success," *IEEE Software*, vol. 12, no. 1, pp. 19–23, jan 1995.

[2] R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen, and J. Hage, "How do professionals perceive legacy systems and software modernization?" in *36th International Conference on Software Engineering*, 2014, pp. 36–47.

[3] GAO, *Information Technology: Agencies Need to Develop Modernization Plans for Critical Legacy Systems*, 2019, https://www.gao.gov/products/gao-19-471.

[4] R. Morris, *Keeping old computers going costs government £2.3bn a year, says report*, 2021, https://www.bbc.com/news/uk-politics-58085316.

[5] D. Beach, *Legacy systems push up costs of digital transformation*, 2019, https://www.theglobaltreasurer.com/2018/09/27/legacy-systems-push-up-costs-of-digital-transformation/.

[6] R. C. Seacord, D. Plakosh, and G. A. Lewis, *Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices*. USA: Addison-Wesley, 2003.

[7] D. Wolfart, W. K. G. Assunção, I. F. da Silva, D. C. P. Domingos, E. Schmeing, G. L. D. Villaca, and D. d. N. Paza, "Modernizing legacy systems with microservices: A roadmap," in *25th Evaluation and Assessment in Software Engineering (EASE)*. ACM, 2021, p. 149–159.

[8] P. L. Leon and F. E. A. Horita, "On the modernization of systems for supporting digital transformation: A research agenda," in *XVII Brazilian Symposium on Information Systems*, 2021, pp. 1–8.

[9] S. Strobl, M. Bernhart, and T. Grechenig, "Towards a topology for legacy system migration," in *IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 586–594.

[10] A. M'baya, J. Laval, and N. Moalla, "An assessment conceptual framework for the modernization of legacy systems," in *11th International Conference on Software, Knowledge, Information Management and Applications*. IEEE, 2017, pp. 1–11.

[11] A. F. Iosif-Lazar, A. S. Al-Sibahi, A. S. Dimovski, J. E. Savolainen, K. Sierszecki, and A. Wasowski, "Experiences from designing and validating a software modernization transformation (e)," in *30th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2015, pp. 597–607.

[12] T. C. Fanelli, S. C. Simons, and S. Banerjee, "A systematic framework for modernizing legacy application systems," in *23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1. IEEE, 2016, pp. 678–682.

[13] R. Khadka, P. Shrestha, B. Klein, A. Saeidi, J. Hage, S. Jansen, E. van Dis, and M. Bruntink, "Does software modernization deliver what it aimed for? a post modernization analysis of five software modernization case studies," in *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2015, pp. 477–486.

[14] R. Seacord, S. Comella-Dorda, G. Lewis, P. Place, and D. Plakosh, "Legacy system modernization strategies," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2001-TR-025, 2001. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5729

[15] C.-C. Chiang and C. Bayrak, "Legacy software modernization," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 2006, pp. 1304–1309.

[16] B. A. Ekanem and E. Woherem, "Dealing with components reusability issues as cutting-edge applications turn legacy," in *SAI Computing Conference (SAI)*. IEEE, 2016, pp. 1190–1198.

[17] A. Khalilipour, M. Challenger, M. Onat, H. Gezgen, and G. Kardas, "Refactoring legacy software for layer separation," *International Journal of Software Engineering and Knowledge Engineering*, vol. 31, no. 02, pp. 217–247, 2021.

[18] A. M. AlSobeh and A. A. Magableh, "An aspect-oriented with bip components for better crosscutting concerns modernization in iot applications," in *CS & IT Conference Proceedings*, vol. 8, no. 12. CS & IT Conference Proceedings, 2018.

[19] S. Rizvi, Z. Khanam, and J. M. Islamia, "A comparative study of using object oriented approach and aspect oriented approach for the evolution of legacy system," *International Journal of Computer Applications*, vol. 975, p. 8887, 2010.

[20] N. Goel, "Legacy systems towards aspect-oriented systems," in *Achieving Enterprise Agility through Innovative Software Development*. IGI Global, 2015, pp. 262–286.

[21] W. K. G. Assunção, R. E. Lopez-Herrejon, L. Linsbauer, S. R. Vergilio, and A. Egyed, "Reengineering legacy applications into software product lines: a systematic mapping," *Empirical Software Engineering*, vol. 22, no. 6, pp. 2972–3016, feb 2017.

[22] J. Åkesson, S. Nilsson, J. Krüger, and T. Berger, "Migrating the android apo-games into an annotation-based software product line," in *23rd International Systems and Software Product Line Conference - Volume A*, ser. SPLC '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 103–107. [Online]. Available: https://doi.org/10.1145/3336294.3342362

[23] J. Krüger, W. Mahmood, and T. Berger, "Promote-pl: A round-trip engineering process model for adopting and evolving product lines," in *24th ACM Conference on Systems and Software Product Line: Volume A - Volume A*, ser. SPLC '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3382025.3414970

[24] W. K. G. Assunção, J. Krüger, and W. D. F. Mendonça, "Variability management meets microservices: Six challenges of re-engineering microservice-based webshops," in *24th ACM Conference on Systems and Software Product Line: Volume A - Volume A*, ser. SPLC '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3382025.3414942

[25] L. Carvalho, A. Garcia, W. K. G. Assunção, R. de Mello, and M. J. de Lima, "Analysis of the criteria adopted in industry to extract microservices," in *7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice*. IEEE, 2019, pp. 22–29.

[26] P. Di Francesco, P. Lago, and I. Malavolta, "Migrating towards microservice architectures: an industrial survey," in *International conference on software architecture*. IEEE, 2018, pp. 29–2909.

[27] Y. Wang, H. Kadiyala, and J. Rubin, "Promises and challenges of microservices: an exploratory study," *Empirical Software Engineering*, vol. 26, no. 4, may 2021.

[28] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, 2017.

[29] H. Knoche and W. Hasselbring, "Using microservices for legacy software modernization," *IEEE Software*, vol. 35, no. 3, pp. 44–49, 2018.

[30] S. M. Salman, A. V. Papadopoulos, S. Mubeen, and T. Nolte, "A systematic methodology to migrate complex real-time software systems to multi-core platforms," *Journal of Systems Architecture*, vol. 117, p. 102087, 2021.

[31] V. T. R and A. A. Chikkamannur, "A methodology for migration of software from single-core to multi-core machine," in *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, 2016, pp. 367–369.

[32] C. Norton, C. Zuffada, O. Kalashnikova, and V. Decyk, "Challenges in modernizing legacy scientific software," in *AGU Fall Meeting Abstracts*, vol. 2008, 2008, pp. IN11A–1016.

[33] R. Pérez-Castillo, M. A. Serrano, and M. Piattini, "Software modernization to embrace quantum technology," *Advances in Engineering Software*, vol. 151, p. 102933, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997820309790

[34] N. Bommadevara, A. Del Miglio, and S. Jansen, "Cloud adoption to accelerate it modernization," *Digitial McKinsey: Insights*, 2018.

[35] R. Cherinka, S. Foote, J. Burgo, and J. Prezzama, "The impact of agile methods and "devops" on day 2+ operations for large enterprises," in *Intelligent Computing*, K. Arai, Ed. Cham: Springer International Publishing, 2022, pp. 1068–1081.

[36] N. C. Mendonça, C. Box, C. Manolache, and L. Ryan, "The monolith strikes back: Why istio migrated from microservices to a monolithic architecture," *IEEE Software*, vol. 38, no. 5, pp. 17–22, 2021.

[37] Comella-Dorda, Wallnau, Seacord, and Robert, "A survey of black-box modernization approaches for information systems," in *International Conference on Software Maintenance*, 2000, pp. 173–183.

[38] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann, "Microservices migration in industry: Intentions, strategies, and challenges," in *International Conference on Software Maintenance and Evolution*. IEEE, 2019, pp. 481–490.

[39] P. Robertson, "Integrating legacy systems with modern corporate applications," *Commun. ACM*, vol. 40, no. 5, p. 39–46, May 1997. [Online]. Available: https://doi.org/10.1145/253769.253785

[40] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[41] G. K. Michelon, W. K. G. Assunção, D. Obermann, L. Linsbauer, P. Grünbacher, and A. Egyed, "The life cycle of features in highly-configurable software systems evolving in space and time," in *20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*. ACM, 2021.

[42] C. H. Costa, P. H. M. Maia, N. C. Mendonça, and L. S. Rocha, "Supporting partial database migration to the cloud using non-intrusive software adaptations: An experience report," in *Advances in Service-Oriented and Cloud Computing*, A. Celesti and P. Leitner, Eds. Cham: Springer International Publishing, 2016, pp. 238–248.

[43] K. Prokofyev, O. Dmitrieva, T. Zmyzgova, and E. Polyakova, "Modern engineering education as a key element of russian technological modernization in the context of digital economy," in *International Scientific Conference "Far East Con"(ISCFEC 2018) Advances in Economics, Business and Management Research*, vol. 47, 2019, pp. 652–656.

[44] M.-L. Sánchez-Gordón, R. Colomo-Palacios, A. de Amescua Seco, and R. V. O'Connor, *The Route to Software Process Improvement in Small- and Medium-Sized Enterprises*. Cham: Springer International Publishing, 2016, pp. 109–136.

[45] I. F. da Silva, P. A. da Mota Silveira Neto, P. O'Leary, E. S. de Almeida, and S. R. de Lemos Meira, "Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study," *Journal of Systems and Software*, vol. 88, pp. 189–206, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121213002598

[46] B. Althani, S. Khaddaj, and B. Makoond, "A quality assured framework for cloud adaptation and modernization of enterprise applications," in *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, 2016, pp. 634–637.