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

Wesley K. G. Assunção
North Carolina State University
Raleigh, USA

Luciano Marchezan
Alexander Egyed
Johannes Kepler University
Linz, Austria

Rudolf Ramler
Software Competence Center
Hagenberg GmbH (SCCH)
Hagenberg, Austria

ABSTRACT

Software modernization is an inherent activity of software engineering, as technology advances and systems inevitably become outdated. The term "software modernization" emerged as a research topic in the early 2000s, with a differentiation from traditional software evolution. Studies on this topic became popular due to new programming paradigms, technologies, and architectural styles. Given the pervasive nature of software today, modernizing legacy systems is paramount to provide users with competitive and innovative products and services. Despite the large amount of work available in the literature, there are significant limitations: (i) proposed approaches are strictly specific to one scenario or technology, lacking flexibility; (ii) most of the proposed approaches are not aligned with the current modern software development scenario; and (iii) due to a myriad of proposed modernization approaches, practitioners may be misguided on how to modernize legacies. In this work, our goal is to call attention to the need for advances in research and practices toward a well-defined software modernization domain. The focus is on enabling organizations to preserve the knowledge represented in legacy systems while taking advantages of disruptive and emerging technologies. Based on this goal, we put the different perspectives of software modernization in the context of contemporary software development. We also present a research agenda with 10 challenges to motivate new studies.

CCS CONCEPTS

• Software and its engineering → Software evolution; **Software architectures**; **• Computer systems organization → Cloud computing**.

KEYWORDS

Software migration, re-engineering, research agenda

ACM Reference Format:
Wesley K. G. Assunção, Luciano Marchezan, Alexander Egyed, and Rudolf Ramler. 2024. Contemporary Software Modernization: Perspectives and Challenges to Deal with Legacy Systems. In *Proceedings of International Workshop on Software Engineering in 2030 (SE 2030)*. ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Throughout the life of a software system, its architecture decays, its underlying technologies become obsolete, the user requirements change, or the company's business models evolve—ultimately, causing the software to morph into what we call legacy systems [8]. The large majority of software currently in use are long-lived systems that represent many years of competitive knowledge and business value [26]. 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 [7, 23, 36]. 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 [23]. 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 [36].

To remain competitive, companies must modernize their legacy systems, preserving the hard-earned knowledge acquired through many years of system development [26, 48, 52]. According to Seacord et al. [48] "*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 [48]. Furthermore, modernization is a mean to leverage the digital transformation [7], as it enables the use of emerging/disruptive technologies such as artificial intelligence, high-performance computing, cloud computing, IoT, robotics, and big data [31].

In the literature, we can find different modernization strategies [31, 49]. 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/GPUs devices. Even the software development process has been modernized, e.g., agile methods [32] and DevOps [13]. Additionally, modernization has different driving forces and impacts related to *organizational*, *operational*, and *technological* aspects [52]. 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 segments [49, 52].

Despite the existing studies on the topic of modernization, covering different strategies and aspects, there are still significant limitations and gaps in the state of the art and the practice: (i) existing approaches are too specific and typically imply to individual technologies or specific modernization scenarios only, without flexibility. This limits their usefulness, reusability, or adaptation for different technologies or scenarios; (ii) proposed approaches typically are not aligned with each other—often providing fragments of modernization and at times even becoming outdated, as there is no contemporaneous body of knowledge on the fundamentals

SE 2030, July 2024, Porto de Galinhas, Brazil
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

of software modernization; and (iii) the existence of several 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 studies that try to organize the existing pieces of work on software modernization have several limitations. They only present an overview of the state of the art [31]; are based on few case studies or a subset of existing literature [25, 33, 49]; are outdated regarding current emerging/disruptive technologies [20, 25–27]; partially cover the modernization life cycle, and rarely take into account organizational, operational, and technological aspects [31, 52]. As pieces of work span across many years and focus on modernizing for different purposes, there is a need for discussing modernization in the context of the contemporaneous software development.

In this work, our goal is to call attention to the need for advances in research and practices towards software modernization in the light of contemporary software development. The focus is to preserve knowledge represented in legacy systems while employing disruptive and emerging technologies to the benefit of users, companies, and society. Based on that, we contextualize the different perspectives of software modernization and introduce a research agenda with 10 challenges to be taken into account. Our contribution is to motivate the discussion on software modernization, present open challenges, and alert companies about risks in adopting solutions based on popularity or hypes.

2 BACKGROUND AND RELATED WORK

Seacord et al. [48] presented software modernization as a remedy to face the legacy system crisis in the early 2000s. They discussed how to keep or add business value through modern technologies, reducing operational costs, and dealing with technical aspects, e.g., allowing better reuse and easier maintenance [48]. 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. [48] to bring forward a contemporaneous perspective of software modernization. In addition to the *technical quality* and *business value* dimensions, we introduce *innovation* as additional dimension that is achieved by new disruptive and emerging technologies—driving forces 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 undergoing a re-engineering or migration process.
- **2 Maintain**: systems with high technical quality and low business value should not require modernization effort, but traditional maintenance activities should be used, 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.

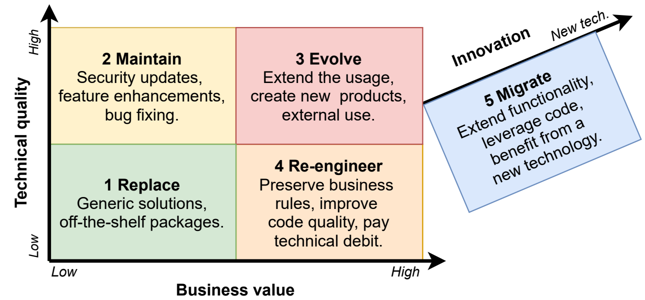

Figure 1: Extended quadrant of the portfolio analysis for the contemporary software modernization, adapted from [47].

- **4 Re-engineer**: systems with high business value and low technical quality should be re-engineered in order to preserve business value, i.e., external quality, and manage the technical debt, i.e., internal quality. This type of modernization can be transparent to the end user.
- ***5 Migrate***: when the system has high business value and a company decides to drive innovation with emerging or disruptive technologies, independently of the system's technical quality, a migration to the desired new technologies should take place. This is, for example, the case when companies foster a digitalization initiative.

In the literature, we can find *several modernization strategies* to retain business value of legacy systems [31, 49]. For example, restructuring systems using components [14, 19, 28]; adoption of aspect-oriented development [3, 24, 43]; re-engineering of system variants into software product lines [5, 6, 30, 42]; migration to microservices [10, 12, 18, 29, 50–52]; supporting new devices or pieces of hardware, e.g., from single-core to multi-core machines [38, 41, 45]; classical information systems to quantum computing [40, 53]; and leveraging the use of AI/ML/Foundation Models [1], a current trend. Even the software development process has been modernized, e.g., agile methods [32] and DevOps [9, 13]. Also, modernization has different driving forces and impacts related to *organizational*, *operational*, and *technological* aspects. The modernization can focus on independence of teams, optimizing deployment, adding innovation, facilitating scalability, or exploring new market segment [49, 52].

Despite existing literature, the work on software modernization has several limitations: studies only present an overview of the state of the art [31]; are based on few case studies or a subset of existing literature [25, 33, 49]; are outdated regarding current emerging/disruptive technologies [20, 25–27]; partially cover the modernization life cycle, and rarely take into account *organizational*, *operational*, and *technological* aspects [31, 52]. Furthermore, these studies are limited to exploring contemporary needs, e.g., digital transformation [31]. Finally, software modernization must be seen as a multi-perspective activity, which is discussed next.

3 MULTI-PERSPECTIVE AND CHALLENGES

In this paper, we propose a multi-perspective of software modernization in the context of contemporary software development. Figure 2 presents six perspectives that affect the process of modernizing a legacy system. These perspectives range from understanding the legacy system, to conducting the transition from the legacy (or part of) to the modern system. Based on this multi-perspective, together with known needs, trends, and recent pieces of work in the topic of software modernization, we present a research agenda structured as a list of challenges (C), described in what follows.

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 2. Based on that, there is a need for a comprehensive and contemporaneous body of knowledge about modernization strategies. We do not have to reinvent the wheel but organize existing knowledge in the light of the perspectives presented in Figure 2 and contemporary software development approaches. Thus, we propose that researchers should gather and classify a body of knowledge on software modernization based on both research and practice. For example, a first step in that direction could be the knowledge base of architecture decision records collected from open source projects in [11].

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 modernization. Based on specific goals, some approaches are more appropriate than others. However, this recommendation must be an informed decision. The challenge here is to provide guidelines to support practitioners and companies on how to choose the proper approach according to their goals, avoiding deciding only based on technology "hypes". For example, microservice-based architectures have 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 [52]. Choosing the wrong approach can lead to inefficiency and frustration in modernization. Hence, systems transformed into microservice were migrated back to monolithic applications [34]. Based on such experiences, future research should compile a body of knowledge (C_1) to derive guidelines for recommending and adopting customizable or domain-specific modernization approaches.

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 [15], which is the replacement of the legacy system with the modern one at once; (ii) incremental modernization, following a strangler pattern [22], in which parts of the legacy systems are incrementally replaced by modern parts [48, 52]; and (iii) the coexistence, in which legacy and modern parts operate together in one system [44]. 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. In this context, we envision research to investigate how to enable hybrid environments, focusing on the coexistence of legacy and modern. This is related to C_5, as the decision made with regard to what to do with the legacy 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 [5, 18]. However, modernization has different driving forces and impacts related to *organizational*, *operational*, and *technological* aspects [47–49, 52]. Software engineering involves technologies, people, and processes to be aligned with business strategies [21]. The challenge here is to propose approaches that deal with all these aspects. While technical aspects must always be considered, we argue that future work has to further explore organizational and operational aspects alongside the technical ones, balancing their priority and cost-benefit.

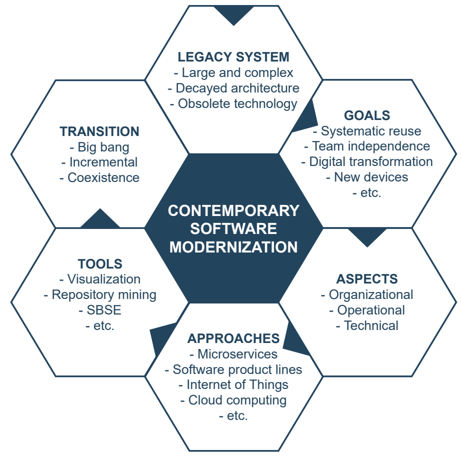

Figure 2: Different perspectives of software modernization in the context of contemporary software development.

C_5: Decide among replace, maintain, evolve, re-engineer, 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. For that, we expect future work to propose recommendation approaches for decision-making support in terms of modernization possibilities, also taking into account organizational, operational, and technological aspects (C_4).

C_6: Support digital transformation. Digital transformation is currently a trend and is receiving great attention around the world. For example, the European Union has the Digital Europe Programme,[1] Australia has the Digital Economy Strategy,[2] in North America there are 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, the digital transformation is hampered by legacy systems [7]. In this context, modernization is a means to leverage the digital transformation [7, 31]. 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 overview. In this direction, we envision future work

[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/

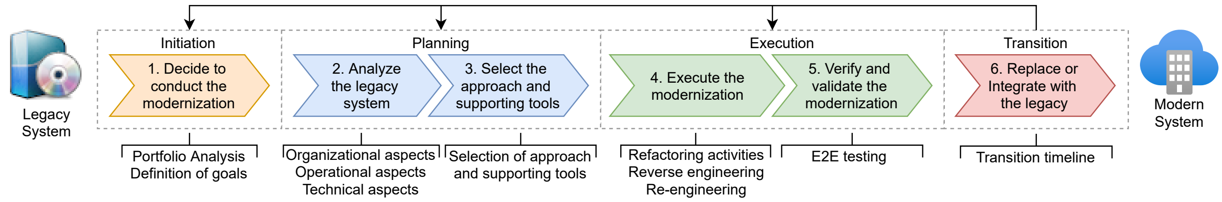

Figure 3: Preliminary multi-perspective modernization workflow in the context of contemporary software development.

enriching the legacy system with modern emerging and disruptive technologies to create new services and operations to companies' workforce and users.

C_7: Prepare the legacy for the modernization. When the legacy system has a high business value, it is a candidate for 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 features being revised [35], 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. This relates to the possibility regarding hybrid environments (C_3) as during the preparation for the evolution, the old legacy and the new migrated system may have to coexist. A research direction is to leverage Foundation Models in tasks related to code understanding [37] and refactoring [2] during the modernization.

C_8: Propose non-intrusive approaches and techniques. Practitioners usually have preferences for using some technologies, tools, and workflows. Based on that, researchers should propose modernization approaches and tools that take into account these preferences. Non-intrusive approaches and techniques are easier to transfer to practice [16]. Thus, we envisage that in addition to propose new solutions to the modernization challenges (e.g., C_3, C_5, C_6 and C_7), researches should also think of lightweight ways of integrating such solution in the technologies, tools, and workflows already in use by practitioners. This challenge is related to C_4, as the need for non-intrusive approaches reflects the need to consider operational and organizational aspects of companies.

C_9: Train workforce with skills for dealing with modernization. Figure 2 presents the different perspectives of the software modernization. These several perspectives must be considered to train the workforce in charge of operationalizing the modernization process [39]. Thus, a challenge is to training the workforce with expertise to deal with the complexity of software modernization [40]. To address this challenges, educators can benefit from contributions to C_1, in which a body of knowledge can serve as the basis for designing new courses in academia. Training also relates to C_4, as it directly affects operational and organizational aspects that are important for companies, such as empowering employees.

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 [17, 46]. This might also be the case for software modernization [4]. Based on that, research needs to be conducted to deal with challenges faced by SMEs when modernizing their legacy systems to grow and be more competitive. This should be considered by researchers when creating a body of knowledge for software modernization. The modernization in context of SMEs also relate directly to C_4, as the organizational aspects of SMEs are different from big enterprises. Consequently, it may have a significant impact on the decision to modernize the legacy or not (C_5).

4 THE MODERNIZATION WORKFLOW

Based on recent systematic mapping studies [5, 52], we defined a preliminary multi-perspective modernization workflow, which is presented in Figure 3. This process is composed of four phases, namely initiation, planning, execution, and transition. Additionally, these phases have six activities. The activities are sequential, but it is possible to return to previous activities, as shown by the arrows. Below each activity, we describe some tasks, which are related to the modernization quadrant (Figure 1) and the multi perspectives (Figure 2). This workflow is an initial proposal for establishing a general process to be enriched with specific information or additional activities to deal with the challenges presented in Section 3. For instance, if a process is desired to SMEs (C_{10}), constraints related to limited resources should be considered.

5 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 research 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. These challenges can be employed as a research agenda for future work on this topic.

ACKNOWLEDGMENTS

The research reported in this paper has been funded by BMK, BMDW, and the State of Upper Austria in the frame of SCCH, part of the COMET Programme managed by FFG as well as the Austrian Science Fund (FWF, P31989-N31).

REFERENCES

[1] Shivali Agarwal, Sridhar Chimalakonda, Saravanan Krishnan, Vini Kanvar, and Samveg Shah. 2024. Tutorial Report on Legacy Software Modernization: A Journey From Non-AI to Generative AI Approaches. In *17th Innovations in Software Engineering Conference (ISEC)*. ACM, Article 19, 3 pages. https://doi.org/10.1145/3641399.3641434

[2] Eman Abdullah AlOmar, Anushkrishna Venkatakrishnan, Mohamed Wiem Mkaouer, Christian D Newman, and Ali Ouni. 2024. How to Refactor this Code? An Exploratory Study on Developer-ChatGPT Refactoring Conversations. *arXiv preprint arXiv:2402.06013* (2024).

[3] Anas MR AlSobeh and Aws A Magableh. 2018. An Aspect-Oriented With BIP Components for Better Crosscutting Concerns Modernization in IOT Applications. In *CS & IT Conference Proceedings*, Vol. 8. CS & IT Conference Proceedings.

[4] B. Althani, S. Khaddaj, and B. Makoond. 2016. A Quality Assured Framework for Cloud Adaptation and Modernization of Enterprise Applications. In *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)*. 634–637. https://doi.org/10.1109/CSE-EUC-DCABES.2016.251

[5] Wesley K. G. Assunção, Roberto E. Lopez-Herrejon, Lukas Linsbauer, Silvia R. Vergilio, and Alexander Egyed. 2017. Reengineering legacy applications into software product lines: a systematic mapping. *Empirical Software Engineering* 22, 6 (feb 2017), 2972–3016. https://doi.org/10.1007/s10664-017-9499-z

[6] Wesley K. G. Assunção, Jacob Krüger, and Willian D. F. Mendonça. 2020. 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* (Montreal, Quebec, Canada) *(SPLC '20)*. Association for Computing Machinery, New York, NY, USA, Article 22, 6 pages. https://doi.org/10.1145/3382025.3414942

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

[8] K. Bennett. 1995. Legacy systems: coping with success. *IEEE Software* 12, 1 (jan 1995), 19–23. https://doi.org/10.1109/52.363157

[9] Nagendra Bommadevara, Andrea Del Miglio, and Steve Jansen. 2018. Cloud adoption to accelerate IT modernization. *Digitial McKinsey: Insights* (2018).

[10] Georg Buchgeher, Rudolf Ramler, Heinz Stummer, and Hannes Kaufmann. 2021. Adopting Microservices for Industrial Control Systems: A Five Step Migration Path. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–8.

[11] Georg Buchgeher, Stefan Schöberl, Verena Geist, Bernhard Dorninger, Philipp Haindl, and Rainer Weinreich. 2023. Using Architecture Decision Records in Open Source Projects–An MSR Study on GitHub. *IEEE Access* (2023).

[12] Luiz Carvalho, Alessandro Garcia, Wesley K. G. Assunção, Rafael de Mello, and Maria Julia de Lima. 2019. 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, 22–29.

[13] R. Cherinka, S. Foote, J. Burgo, and J. Prezzama. 2022. The Impact of Agile Methods and "DevOps" on Day 2+ Operations for Large Enterprises. In *Intelligent Computing*, Kohei Arai (Ed.). Springer, Cham, 1068–1081.

[14] Chia-Chu Chiang and Coskun Bayrak. 2006. Legacy Software Modernization. In *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 2. 1304–1309. https://doi.org/10.1109/ICSMC.2006.384895

[15] Comella-Dorda, Wallnau, Seacord, and Robert. 2000. A survey of black-box modernization approaches for information systems. In *International Conference on Software Maintenance*. 173–183. https://doi.org/10.1109/ICSM.2000.883039

[16] Caio H. Costa, Paulo H. M. Maia, Nabor C. Mendonça, and Lincoln S. Rocha. 2016. Supporting Partial Database Migration to the Cloud Using Non-intrusive Software Adaptations: An Experience Report. In *Advances in Service-Oriented and Cloud Computing*. Springer, Cham, 238–248.

[17] Ivonei Freitas da Silva, Paulo Anselmo da Mota Silveira Neto, Pádraig O'Leary, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. 2014. Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study. *Journal of Systems and Software* 88 (2014), 189–206. https://doi.org/10.1016/j.jss.2013.10.040

[18] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2018. Migrating towards microservice architectures: an industrial survey. In *International conference on software architecture*. IEEE, 29–2909.

[19] Bassey Asuquo Ekanem and Evans Woherem. 2016. Dealing with components reusability issues as cutting-edge applications turn legacy. In *SAI Computing Conference (SAI)*. IEEE, 1190–1198. https://doi.org/10.1109/SAI.2016.7556129

[20] Timothy C Fanelli, Scott C Simons, and Sean Banerjee. 2016. A systematic framework for modernizing legacy application systems. In *23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. IEEE, 678–682.

[21] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176–189. https://doi.org/10.1016/j.jss.2015.06.063

[22] Jonas Fritzsch, Justus Bogner, Stefan Wagner, and Alfred Zimmermann. 2019. Microservices Migration in Industry: Intentions, Strategies, and Challenges. In *International Conference on Software Maintenance and Evolution*. IEEE, 481–490.

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

[24] Noopur Goel. 2015. Legacy Systems towards Aspect-Oriented Systems. In *Achieving Enterprise Agility through Innovative Software Development*. IGI Global, 262–286.

[25] Alexandru F Iosif-Lazar, Ahmad Salim Al-Sibahi, Aleksandar S Dimovski, Juha Erik Savolainen, Krzysztof Sierszecki, and Andrzej Wasowski. 2015. Experiences from designing and validating a software modernization transformation (E). In *30th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 597–607.

[26] Ravi Khadka, Belfrit V Batlajery, Amir M Saeidi, Slinger Jansen, and Jurriaan Hage. 2014. How do professionals perceive legacy systems and software modernization?. In *36th International Conference on Software Engineering*. 36–47.

[27] Ravi Khadka, Prajan Shrestha, Bart Klein, Amir Saeidi, Jurriaan Hage, Slinger Jansen, Edwin van Dis, and Magiel Bruntink. 2015. 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, 477–486.

[28] Alireza Khalilipour, Moharram Challenger, Mehmet Onat, Hale Gezgen, and Geylani Kardas. 2021. Refactoring Legacy Software for Layer Separation. *International Journal of Software Engineering and Knowledge Engineering* 31, 02 (2021), 217–247.

[29] Holger Knoche and Wilhelm Hasselbring. 2018. Using Microservices for Legacy Software Modernization. *IEEE Software* 35, 3 (2018), 44–49. https://doi.org/10.1109/MS.2018.2141035

[30] Jacob Krüger, Wardah Mahmood, and Thorsten Berger. 2020. Promote-Pl: A Round-Trip Engineering Process Model for Adopting and Evolving Product Lines. In *24th ACM Conference on Systems and Software Product Line (SPLC)*. ACM, Article 2, 12 pages. https://doi.org/10.1145/3382025.3414970

[31] Pablo Luiz Leon and Flávio Eduardo Aoki Horita. 2021. On the modernization of systems for supporting digital transformation: A research agenda. In *XVII Brazilian Symposium on Information Systems*. 1–8.

[32] Robert Cecil Martin. 2003. *Agile software development: principles, patterns, and practices.* Prentice Hall PTR.

[33] Abir M'baya, Jannik Laval, and Nejib Moalla. 2017. An assessment conceptual framework for the modernization of legacy systems. In *11th International Conference on Software, Knowledge, Information Management and Applications*. IEEE, 1–11.

[34] Nabor C. Mendonça, Craig Box, Costin Manolache, and Louis Ryan. 2021. The Monolith Strikes Back: Why Istio Migrated From Microservices to a Monolithic Architecture. *IEEE Software* 38, 5 (2021), 17–22. https://doi.org/10.1109/MS.2021.3080335

[35] Gabriela K. Michelon, Wesley K. G. Assunção, David Obermann, Lukas Linsbauer, Paul Grünbacher, and Alexander Egyed. 2021. 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.

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

[37] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an llm to help with code understanding. In *IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE, 881–881.

[38] CD Norton, C Zuffada, OV Kalashnikova, and VK Decyk. 2008. Challenges in Modernizing Legacy Scientific Software. In *AGU Fall Meeting Abstracts*, Vol. 2008. IN11A–1016.

[39] KG Prokofyev, OV Dmitrieva, TR Zmyzgova, and EN Polyakova. 2019. 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. 652–656.

[40] Ricardo Pérez-Castillo, Manuel A. Serrano, and Mario Piattini. 2021. Software modernization to embrace quantum technology. *Advances in Engineering Software* 151 (2021), 102933. https://doi.org/10.1016/j.advengsoft.2020.102933

[41] Vinay T R and Ajeet A Chikkamannur. 2016. 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)*. 367–369. https://doi.org/10.1109/CSITSS.2016.7779388

[42] Jonas Åkesson, Sebastian Nilsson, Jacob Krüger, and Thorsten Berger. 2019. Migrating the Android Apo-Games into an Annotation-Based Software Product Line. In *23rd International Systems and Software Product Line Conference (SPLC)*. ACM, 103–107. https://doi.org/10.1145/3336294.3342362

[43] SAM Rizvi, Zeba Khanam, and Jamia Millia Islamia. 2010. A Comparative Study of using Object oriented approach and Aspect oriented approach for the Evolution

of Legacy System. *International Journal of Computer Applications* 975 (2010), 8887.

[44] Paul Robertson. 1997. Integrating Legacy Systems with Modern Corporate Applications. *Commun. ACM* 40, 5 (May 1997), 39–46. https://doi.org/10.1145/253769.253785

[45] Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. 2021. A systematic methodology to migrate complex real-time software systems to multi-core platforms. *Journal of Systems Architecture* 117 (2021), 102087. https://doi.org/10.1016/j.sysarc.2021.102087

[46] Mary-Luz Sánchez-Gordón, Ricardo Colomo-Palacios, Antonio de Amescua Seco, and Rory V. O'Connor. 2016. *The Route to Software Process Improvement in Small- and Medium-Sized Enterprises*. Springer, Cham, 109–136. https://doi.org/10.1007/978-3-319-31545-4_7

[47] Robert Seacord, Santiago Comella-Dorda, Grace Lewis, Patrick Place, and Daniel Plakosh. 2001. *Legacy System Modernization Strategies*. Technical Report CMU/SEI-2001-TR-025. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5729

[48] Robert C. Seacord, Daniel Plakosh, and Grace A. Lewis. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices*. Addison-Wesley, USA.

[49] Stefan Strobl, Mario Bernhart, and Thomas Grechenig. 2020. Towards a Topology for Legacy System Migration. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 586–594.

[50] Davide Taibi, Valentina Lenarduzzi, and Claus Pahl. 2017. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing* 4, 5 (2017), 22–32.

[51] Yingying Wang, Harshavardhan Kadiyala, and Julia Rubin. 2021. Promises and challenges of microservices: an exploratory study. *Empirical Software Engineering* 26, 4 (may 2021). https://doi.org/10.1007/s10664-020-09910-y

[52] Daniele Wolfart, Wesley K. G. Assunção, Ivonei F. da Silva, Diogo C. P. Domingos, Ederson Schmeing, Guilherme L. Donin Villaca, and Diogo do N. Paza. 2021. Modernizing Legacy Systems with Microservices: A Roadmap. In *25th Evaluation and Assessment in Software Engineering (EASE)*. ACM, 149–159.

[53] Xudong Zhao, Xiaolong Xu, Lianyong Qi, Xiaoyu Xia, Muhammad Bilal, Wenwen Gong, and Huaizhen Kou. 2024. Unraveling quantum computing system architectures: An extensive survey of cutting-edge paradigms. *Information and Software Technology* 167 (2024), 107380.