Addis Ababa Science and Technology University

College of Engineering

Department of Software Engineering

Software Evolution Maintenance

Success Factors for Software Reuse

Name	ID	Section
Metsehafe Eyasu	ETS0852/13	C
Michael Engida	ETS0859/13	C
Mohammed Mehad	ETS0907/13	C
Mohammed Restem	ETS0911/13	C
Mohammed Shemim	ETS0912/13	C

Technical Success Factors for Software Reuse: Recent Findings

Introduction

Software reuse involves leveraging existing software assets to enhance software quality, productivity, and overall efficiency [1]. Recent research highlights that systematic reuse significantly surpasses opportunistic reuse regarding product quality, reliability, productivity, and reduced development time [1] [2]. However, achieving systematic software reuse involves overcoming multiple technical challenges. This summary specifically reviews recent technical factors critical to achieving effective software reuse, based on studies from 2022 to 2025.

Technical Success Factors

Architecture Design and Modularity

Effective architectural practices, particularly modular designs featuring clear, stable interfaces, significantly facilitate software reuse. Microservices architectures inherently support reuse by isolating services into independent, interchangeable modules, simplifying integration, testing, and replacement [2]. Product-line reference architectures explicitly designed with reuse considerations, such as defined variability points and clear delineation of responsibilities, enhance component adaptability across various applications and domains [3].

The principles of low coupling and high cohesion remain pivotal; components designed with these principles are considerably easier and more cost-effective to reuse, modify, and maintain. These principles minimize integration complexity, reduce the likelihood of changes causing widespread disruptions, and streamline long-term software maintenance [4]. Recent case studies confirm that organizations implementing modular architectures experience significantly improved software development timelines and reduced redundancy.

Standardization and Documentation

Adopting standardized coding practices, architectural guidelines, and comprehensive documentation is an essential technical factor in facilitating successful software reuse [3] [5]. Standardization ensures compatibility among diverse components, simplifying integration efforts across development teams and reducing misunderstandings or misalignments.

Detailed, clear, and consistent documentation remains equally vital, as it provides developers with the necessary context, usage guidelines, and precise API details, significantly enhancing the ease and reliability of reusing software assets [5]. Industry-specific standards, such as those used in avionics and healthcare, further promote interoperability, trust, and adherence to quality and safety regulations, facilitating broader reuse adoption [6].

Automation Tools and Platforms

Automated tools and infrastructure significantly reduce reuse friction, improve component quality, and facilitate seamless integration processes. Continuous integration and continuous deployment (CI/CD) automation ensure consistent quality assurance, reducing the risk associated with reused components and promoting developer confidence in reuse initiatives [2].

InnerSource platforms amplify reuse potential by fostering collaborative development environments, enabling visibility and contributions across organizational boundaries. This significantly enhances component discoverability, reduces redundancy, and accelerates innovation through collective knowledge sharing [2]. Additionally, emerging intelligent tooling utilizing machine learning techniques to automatically identify potentially reusable components based on quality metrics, complexity measures, and modularity assessments shows promise for significantly streamlining reuse workflows [7].

Reuse Metrics and Measurement

Defining and systematically tracking reuse-specific metrics such as defect rates, development-time reductions, reuse frequency, component adoption rates, and developer satisfaction levels are critical for quantifying the value and efficacy of reuse initiatives [1] [7]. Metrics serve as crucial feedback mechanisms, informing strategic decision-making, highlighting areas for improvement, and providing tangible evidence to support continued investment in reuse practices.

Organizations employing robust measurement frameworks demonstrate greater effectiveness in refining reuse strategies, identifying high-value components, addressing barriers to reuse adoption, and driving continuous improvement cycles. Recent research highlights that organizations with well-defined measurement strategies consistently report higher reuse maturity levels and greater overall success.

Domain Engineering and Systematic Reuse

Systematic approaches to reuse, notably Domain Engineering and Software Product Line Engineering (SPLE), are fundamental to successful long-term reuse strategies. These methods systematically analyze, design, and manage reusable assets tailored specifically for application families or business domains [3] [8].

Comprehensive domain-specific repositories containing rigorously defined and actively managed reusable components are essential components of effective reuse infrastructure. Understanding domain variability, managing product-line architectures, and clearly defining reuse processes ensures components are readily applicable across multiple product variations, significantly improving reuse success rates and overall software quality [3] [8].

Recent case studies further validate that organizations successfully employing domain engineering techniques achieve more rapid development cycles, substantial cost reductions, and higher product quality compared to those relying on opportunistic reuse approaches.

Training and Knowledge Management

While often overlooked, training and knowledge management systems represent crucial supporting technical factors. Organizations investing in comprehensive training programs to enhance developers' reuse skills significantly improve overall adoption and effectiveness. Effective knowledge management systems that capture reuse experiences, lessons learned, and best practices further reinforce organizational reuse culture and capability.

Structured onboarding programs, workshops, and ongoing professional development in reuse techniques help sustain long-term success, reduce resistance, and encourage a culture of systematic reuse.

Conclusion

Technical success factors essential for effective software reuse include modular and service-oriented architecture design, robust standardization, detailed documentation, advanced automation tools, well-defined reuse metrics, systematic domain engineering, and comprehensive training and knowledge management practices. Together, these technical elements transform software reuse from an occasional practice to a sustainable, strategic advantage, delivering substantial improvements in quality, productivity, and efficiency across software projects.

References

[1] X. Chen, M. Usman and D. Badampudi, "Understanding and evaluating software reuse costs and benefits from industrial cases—A systematic literature review," *Information and Software Technology,* vol. 171, 2024.

[2] D. Badampudi, M. Usman and X. Chen, "Large scale reuse of microservices using CI/CD and InnerSource practices – a case study," *Empirical Software Engineering,* vol. 30, p. 41, 2025.

[3] W.-T. Lee and C.-H. Chen, "Agile software development and reuse approach with Scrum and software product line engineering," *Electronics,* vol. 12, 2023.

[4] R. F. Gonçalves, C. M. L. Werner and C. M. d. Farias, "Investigating developer experience in software reuse," in *Proc. SBES/SBCARS 2024*, Curitiba, Brazil, 2024.

[5] A. M. El-Halawany, H. K. Elminir and H. El-Bakry, "Improving reuse during the development process for web systems," *Scientific Reports,* vol. 14, 2024.

[6] Q. Sun, "Research on civil airborne software reuse technology," in *Proc. ICIDC 2023*, Nanchang, China, 2023.

[7] M. Y. H. Yeow, C. Y. Chong, M. K. Lim and Y. Y. Yen, "Predicting software reuse using machine learning techniques—A case study on open-source Java software systems," *PLoS ONE,* vol. 20, 2025.

[8] J. Kim, "A case study of domain engineering in software product line development," *Journal of Logistics, Informatics and Service Science,* vol. 9, no. 1, pp. 97-115, 2022.