Peer Response 1: Factors which Influence Reusability

In reply to Aleksandr Vygodchikov

Re: Initial post

by Andrius Busilas - Thursday, 21 March 2024, 8:39 PM

Hi Aleksandr,

Your review of Padhy et al. (2018) regarding reusability factors for object-oriented software presents a comprehensive framework for comprehending and prioritizing elements that affect reusability in software development. The proposed hierarchy in the discussion is logical and well-founded, emphasizing the significant impact of architecture and design-related factors on reusability.

The argumentation in favor of the Architecture Driven Approach (ADP) as the primary factor in facilitating reusability is well-supported. A modular, scalable, and flexible architecture sets the stage for efficient reuse of components across diverse projects by providing a clear organizational structure conducive to interchangeability and adaptability.

Similarly, the discussion rightly highlights the importance of Design Patterns (DP) in promoting reusability. By standardizing the design of software components, design patterns streamline development processes, enhance code maintainability, and facilitate component reuse.

Furthermore, the prioritization of Modular Programming (MIP) as a key factor contributing to reusability is justified. Modularization breaks down complex systems into manageable units, fostering component reuse and enhancing code comprehensibility and maintainability.

The integration of factors such as Algorithm Used in the Program (AP), Documentation in Project (DIP), and Knowledge Requirement (KR) appropriately emphasizes the significance of clear problem-solving strategies, comprehensive documentation, and developer expertise in fostering reusability.

However, the review suggests an area for improvement regarding the prioritization of

Test Cases/Test Design (TCTD) and Requirement Analysis (RA). While these factors are indeed crucial for ensuring the functionality and suitability of reusable components, they might warrant a higher position in the hierarchy given their direct impact on software quality and user satisfaction.

In conclusion, the discussion offers valuable insights into the foundational factors that influence reusability in object-oriented software, providing a structured approach for prioritizing considerations in software development and maintenance.

Initial post

by Aleksandr Vygodchikov - Saturday, 16 March 2024, 6:41 PM

Number of replies: 1

In evaluating the reusability factors listed by Padhy et al. (2018) for object-oriented software, a hierarchy of these factors can be built upon on their foundational impact on software development and maintenance.

- 1. Architecture Driven Approach (ADP): The structure or architecture of a project is fundamental to its reusability. An architecture that is modular, scalable, and flexible facilitates the reuse of components across different projects by providing a clear, high-level structure that supports interchangeability and adaptability.
- 2. Design Patterns (DP): The usage of design patterns promotes reusability by ensuring that software components are designed in a standardized way, making it easier to understand, maintain, and reuse them.
- 3. Modules in the Program (MIP): Modular programming inherently supports reusability by breaking down a project into independent, interchangeable modules that can be reused in different contexts.
- 4. An Algorithm Used in the Program (AP): Algorithms solve specific problems. When an algorithm is designed to be generic and abstract, it can be reused across different projects.
- 5. Documentation in Project (DIP): Comprehensive documentation, including requirement specification analysis, supports reusability by making it easier for new developers to understand the software's functionality, structure, and requirements.
- 6. Knowledge Requirement (KR): The accumulation of knowledge, including experiences, ideas, and reasoning, enhance reusability by providing developers with the necessary understanding to reuse components effectively.
- 7. Test Cases/Test Design (TCTD): Well-designed test cases can be reused across projects to ensure that software components meet the required standards and function correctly in new contexts.
- 8. Models in the Project (MP): Models that accurately represent the project task, including meaningful codes and solutions, can be reused to fast-track the development process in future projects.
- 9. Requirement Analysis (RA): The process of determining user expectations for a new or modified product is crucial for ensuring that reusable components meet the necessary requirements in different projects.

- 10. Service Contracts (SC): Define the interaction between different software components. Clear and well-defined service contracts facilitate reusability by ensuring that components can interact seamlessly when reused in different contexts.
- 11. Used in the Data Project (UD): This factor highlights the importance of data reusability, indicating that data accumulated from previous projects can be reused to achieve targets in new projects.

This prioritization is based on the consideration that structural and design-related factors have a foundational impact on reusability, making it easier to integrate and reuse software components across different projects.

References

Padhy, N., Satapathy, S., & Singh, R. P. (2017) 'State-of-the-Art Object-Oriented Metrics and Its Reusability: A Decade Review', in: Satapathy, S. C., et al. (eds). Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 1. Singapore: Springer Singapore Pte. Limited. Available from: http://ebookcentral.proquest.com/lib/universityofessex-ebooks/detail.action?docID=5217051 [Accessed 16 March 2024].