Collaborative Discuss - Reusability In OOP

Initial Post

Reusability in software design provides for a number of benefits to both designers, stakeholders and users. For designers, it allows for consistency and time saving efficiencies, all of which are key to a successful software design enterprise. That same consistency and efficiency also provides benefits for users and stakeholders, who can be guaranteed of the best product and service due to this. Reflecting on the work of Padhy et al. (2018), who have developed a list of 11 aspects of software design that can be reused in different projects. In order of the most important to the least important, they are hereby listed below.

1. Architecture-Driven Approach (ADP):

A well-defined architecture lays the foundation for scalable, maintainable, and reusable software. This factor ensures that reusability considerations are integrated into the software design from the outset, leading to more efficient reuse of components.

2. Knowledge Requirement (KR):

Knowledge is a fundamental aspect of software development. Reusing knowledge generated during the SDLC can significantly enhance the efficiency of subsequent projects and ensures that valuable insights and experiences are leveraged effectively, leading to better decision-making and problem-solving.

3. Modules in the Program (MIP):

Modularization is a key principle in software engineering, promoting reusability, maintainability, and scalability. This factor also contributes to better organisation and understanding of the software architecture.

4. Reusable Properties Reusability Factor (UD):

The ability to reuse data from previous projects can expedite the development process and improve accuracy. Data reuse minimises redundant efforts and facilitates informed decision-making based on past experiences.

5. Models in the Project (MP):

Models serve as blueprints for software systems, capturing essential aspects of their structure, behaviour, and interactions thus giving developers the clarity to facilitate the analysis and reuse of design artefacts.

6. An Algorithm Used in the Program (AP):

Reusable algorithms can simplify the development process and improve the efficiency and performance of software systems. While important, the priority of this factor may vary depending on the specific requirements and nature of the software being developed.

7. Requirement Analysis (RA):

Effective requirement analysis is crucial for understanding the needs and constraints of stakeholders and designing solutions that meet their expectations. This helps developers to identify opportunities for reusability early in the development process.

8. Design Patterns (DP):

Design patterns provide proven solutions to common design problems, promoting reusability, maintainability, and flexibility.

9. Service Contracts (SC):

Service contracts facilitate effective communication between developers and users, thereby assisting in the reuse of software products.

10. Documentation in Project (DIP):

Documentation is very important for facilitating understanding, maintenance, and reuse of software components, but, documentation may be considered less critical than other factors in terms of directly influencing reusability.

11. Test Cases / Test Design (TCTD):

Test cases and test design are necessary for ensuring the quality of software systems but ranks low in priority when it comes to directly influencing the reusability of a software.

The rationale for the rankings above takes into consideration the scope of each part and it’s timing within the project, as well as the prospect of variations that could impact it in the future. For example, Architecture-Driven Approach is listed as the most important due to how foundational it is for the tasks that follow it. The same thinking applies to Knowledge Requirement, as without the applicable knowledge being reused and shared, tasks would become very labour intensive and inefficient. As the authors themselves have noted, Documentation In Project and Test Cases / Test Design rank lower due to their lack of direct impact on the reusability of software.

References

Padhy, Neelamadhab, Suresh Satapathy, and R. P. Singh. "State-of-the-art object-oriented metrics and its reusability: a decade review." In Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 1, pp. 431-441. Springer Singapore, 2018.

Summary Post

Upon receiving feedback from peers regarding the initial ranking of the core aspects that impact the reusability of a piece of object oriented software, as defined by Padhy et al. (2018), the general consensus is that the initial ranking created aligns with the thoughts of the wider computing community. However, it is possible that upon further research and development, the importance of Test Case and Test Design may have initially been neglected. The ability to run a standard testing procedure in a repeated manner is not only time and labour efficient, but also allows for consistent validation throughout the work completed, thus avoiding potential issues that may be presented during both the deployment and maintenance phases of the software development life cycle.

Reflection also highlights the importance placed on Documentation in Project throughout the software development life cycle, which was initially ranked towards the bottom, mirroring the thoughts of Padhy et al (2018) within their original work. Although it would not shift towards the top of a new ranking of reusability factors, it is important to highlight just how critical solid documentation is to the development of software and object oriented programming projects. The standard models noted within UML allow for a structure that is reusable with minimal changes required within the design process for example, with this standard structure being recognised and reviewed by the industry as a whole, thus adding validation to its application.

Software design can be a labour intensive task, with human and financial resources quickly depleting during the creation of a project, however it is clear that by finding ways to reuse assets and skills from previous projects, these projects can be done in the most efficient way possible for all parties involved.

References

Padhy, Neelamadhab, Suresh Satapathy, and R. P. Singh. "State-of-the-art object-oriented metrics and its reusability: a decade review." In Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 1, pp. 431-441. Springer Singapore, 2018.