Discussion Forum 1 – Factors that influence the reusability of OOP Software

Initial Post:

Padhy et.al (2018) present a detailed approach and research methods that they have used to identify reusable assets in software delivery. Based on professional experience and from implementing software I would categorise the assets below in terms of priority. It is fair to say that this is not set, and the selection and priority will vary between projects.

1. RA: Gathering requirements at the start of a project will capture and document the functional and non-functional needs of the projects which will then drive the decision-making.

2. ADP: Architectural design alongside requirements will help drive the selection of existing/re-usable algorithms or new development.

3. KR: Requirements and architectural review will be considered alongside the knowledge of a re-usable algorithm and if it is usable in the scope of the architecture and answers the requirements.

4. DP: Existing design patterns can be evaluated against the requirements or if there is an over-arching DP for the customer it should be considered to ensure the algorithm fits into any agreed standards or protocols.

5. UDP: The data from previous implementations should be considered. While it may not be the same data for previous implementations it can impact design decisions.

6. AP: By developing a new algorithm based on requirements or design will add to a catalogue of reusable assets and may also answer common issues. Or using an existing algorithm can allow for additions or enhancements.

7. MIP: By developing a modular design (A single or several artefacts e.g. a DLL) will allow for easy distribution of code and executables.

8. DIP: Documentation like design or specification documents should map to requirements to support developers in how to build the algorithm and documentation for existing algorithms should be maintained or updated if required.

9. SC: Once design & documentation is complete the contract should be set with the customer. Will decide on testing approaches and collaboration between developer and customer in issues and delivery schedules.

10. TCTD: The steps up to and including the SC facilitate the creation of testing scripts and test cycles against the system/development.

11. MP: Models while still important are lower priority. The model for the algorithm or more importantly the solution can be fulfilled once all the design is agreed upon, how it fits into the architectural design, how it is tested etc. There may be iterations or changes to be considered in the earlier stages and where the aim is to reduce development time a model would be a final output alongside the solution/design artefacts.

It is worth noting this is very subjective and software delivery can vary from project to project. I have approached this based on my role as a solution architect so a role considering the wider aspects of a project. The company I work in is a COTS software vendor and I act as an overall technical lead. Often these assets form deliverables or milestones in a project for how we deliver our implementation projects in the order I have listed. Internally we have our development teams working on our product but then our customers who work across many industries will often mandate their own priorities as well.

Really interested to hear any feedback from the rest of the group and to discuss further.

References:

Padhy, N., Satapathy, S., & Singh, R.P. (2018) 'State-of-the-Art Object-Oriented Metrics and Its Reusability: A Decade Review', in: Satapathy S., Bhateja V., Das S. (eds) Smart Computing and Informatics. Smart Innovation, Systems and Technologies. 77. Springer.

Summary Post:

In my initial post, I created a list based on the 11 reusability factors described and defined by Padhy et al. (2018). Factors such as requirements, design, and architecture were ranked higher than others based on the approach of each project I work on as they are always commercial, customer-facing engagements so these need to be done upfront for the design of each implementation. Factors such as testing came lower on this not because they are less important but because testing from a functional and user perspective can only really surface after the design has been completed.

Even in projects such as enhancement projects or projects where code may be re-used and after completing initial units 1-3 my thought process would still align with my initial assessment based on the fact that testing from a previous development would become part of the re-usable assets but additional and more specific testing would still become part of that project where you would still gather requirements, design artefacts at the initial stages.

From a professional perspective, there are other factors to consider regarding the methodology. Some projects may mandate Agile which could in turn change the priority or order of factors where testing may become more critical within a sprint but a waterfall model will lend itself to considerable design and build effort up front with a testing cycle towards the end of delivery.

However, it is important to note that all the factors are relevant and mandatory in some cases to deliver a successful project. From a functional perspective, a developer should always be executing and testing so a development test strategy and documentation would become a more relevant factor earlier in the delivery and more discreet as a developer activity to verify that code is tested and structured well

enough that testing can be performed, tracked and verified so would become a key asset in re-using code.

References:

Padhy, N., Satapathy, S., & Singh, R.P. (2018) 'State-of-the-Art Object-Oriented Metrics and Its Reusability: A Decade Review', in: Satapathy S., Bhateja V., Das S. (eds) Smart Computing and Informatics. Smart Innovation, Systems and Technologies. 77. Springer.