Discussion Forum: Factors which they consider influence the reusability of a piece of object-oriented software.

Padhy et.al (2018) present a detailed approach and research methods that they have used to identify re-usable 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 re-usable 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 testing approaches, 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.

10. 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, 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 and 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.