TL Note – Complexity based Project Approach

There is no "one-size-fits-all" approach for projects. The level and type of complexity varies greatly across projects, but making sense of complexity should be a primary driver in choosing an appropriate approach for the project. This TL Note provides some guidance for this critical activity and discusses the consequence of under or overestimating complexity.

What is a sense-making Framework

A sense making framework is a helpful resource, and one example is the Cynefin framework (Ref: https://www.youtube.com/watch?v=N7oz366X0-8). Based on the outcome of applying a sense making framework at a point in time, Teams are likely to see their TechLauncher project as either "Complex" or "Complicated". Briefly, the characteristics and appropriate approaches for these are:

Complicated	The relationship between cause and effect can be understood with some up-front analysis.	Sense-analysis-respond Meaning – select good practices based on expert understanding to address problem and progress the project.
Complex	Relationship between cause and effect can only be understood after the event.	Probe–Sense-Respond Meaning – experiment, assess and use to inform decisions for further progress.

Consequence of Underestimating or Ignoring Complexity

When complexity is not well understood, underestimated, or ignored, the impacts can range from minor to disastrous. Some impacts could be:

- Decisions that underpin implementation cannot be well justified. For example, for a project considered to be "complex", some probing/experimentation may be needed in order to assess outcomes before a subsequent decision can be made. Without the probing the decision is ad hoc, not well informed, and difficult to justify.
- Important factors that should be considered for implementation are omitted. For example, without assessment of probing/experimentation outcomes, important functional or non-functional requirements may be missed.
- Some rework will be required as complexity becomes apparent. For example, if a product is implemented to satisfy the most apparent and expected scenarios without consideration of exceptions and unexpected events, then components will need to be reworked to accommodate further scenarios.
- Projects will completely implode and require major pivot or restart. For example, if a team ignores architecturally significant requirements, there is a good chance that they will "fall off the cliff", and major re-architecting of the solution will be needed.

Consequence of Overestimating Complexity

Key tenets of the TechLauncher Program are to maximise the value produced (including recognised by Stakeholders as delivered) through optimising the use of time and resources. Overestimating complexity most likely results in time spent on things that don't need to be done. While this is perhaps the better side of the coin, it does mean that resource use is not optimised, and value is not maximised. For example, the development of a user interface can usually be considered "complicated", hence requiring selection of a best practice approach with appropriate testing at all stages to ensure smooth implementation and as little rework as possible to achieve an acceptable outcome. The same user interface could be developed applying the approach appropriate for a "complex" problem, but is likely to consume more time and effort through the constant adapting and rework to finally reach acceptance of the same outcome.