Requirements Engineering

Applying Proven Requirements Engineering Methods

It's well known that the implementation and testing of Tactical Data Links and associated military platforms is high cost.

The application of proven requirements engineering methods provides systems engineers with a robust method for fulfilling operator, policy and other stakeholder requirements. But, getting users to articulate their needs can often be a challenging process. No process can pull information out of the void when it doesn't exist, but systems engineering takes a robust and scientific approach to requirements management that cleanly and specifically identifies ambiguities and gaps in stated stakeholder needs.

The best way to get a straight answer is to ask a straight question, and the systems engineering process is very good at generating straight questions.

Working with vague or incomplete requirements doesn't just lead to a risk of building the wrong product, it can also risk building the right product badly. Effective projects run individual management tasks rigorously and efficiently, and the ability to follow a rigorous process is severely hindered by a lack of robust inputs. Problems that arise in this way only multiply over time as knock-on effects are generated and start introducing chaos of their own.

The process begins by identifying what users want in terms of a problem that they need to solve, or an opportunity that they want to pursue. Without yet looking to specific solutions, the first step is to develop the 'operational concept'; what users want the system to do. The context and environment for the system – its basic inputs and outputs – should be understood as clearly as possible while the system as a whole is still being treated as a black box. It is important even at this stage to look past acquisition, towards deployment, configuration management, support and retirement.

In a systems engineering process, only then do you start to formally investigate the sorts of systems which could solve the user's problem. This should begin by generating as many ideas as possible about what should go in the black box, and at this stage should not progress beyond identifying a preferred class of solutions. It has been a long-standing maxim in organisational psychology that the most efficient way to solve a problem is to discuss it for as long as possible before proposing solutions; systems engineering embraces this as a philosophy for the stakeholder relationship.

The next step is to identify as wide a set of stakeholders for the system as possible, and talk to users as directly as you can about what their needs for the system will be. If the low priority or impracticality of these needs isn't trivially obvious, this analysis generally falls into a later stage.

In other words, getting stakeholder needs begins by being as open-minded as you can, in as broad a conversation as possible. From there, the job of the systems engineer becomes that of turning these needs into formalised requirements.

The philosopher of science, Karl Popper, famously said that for a statement to be considered scientific, it must be falsifiable; you have to be able to tell the difference between a world in which the statement is true, and a world in which it is false. Similarly, systems engineers work towards requirements by which it is possible to tell the difference between a system that achieves them, and one that doesn't.

Specifically, this means all requirements have to be individually:

- Clear (concise, limited to one idea, impossible to misinterpret);
- Verifiable (related to a specific, identifiable test of success);
- Functional (describe what is to be done, not how it is to be done);
- Feasible (technically achievable, with acceptable research risks);
- Compliant (compatible with regulatory and governance constraints);
- Traceable (related to specific higher-level requirements and ultimately stakeholder needs);
- Unique (not replicating other requirements); and
- Minimal (describe only "must haves", not "nice to haves"); and taken together they must be:
- Complete (define the system in its entirety); and
- Consistent (not contradicting one another, including not contradicting cost and time requirements).

By forcing requirements to be specified in this way, systems engineers can be thoroughly robust and scientific in developing a system model, and can identify in specific terms the straight questions that need to be asked of stakeholders to define the system properly.

Systems engineers also take a comparably atomised approach to risk; "if [event] then [consequence for stakeholder]". Risk assessment needs to begin at, or before, the requirements phase, and while requirements are being defined, the systems engineer must also look to asking what the consequences of the likelihood of failing to meet those requirements would be.

By treating the requirements engineering process like generating a scientific hypothesis, systems engineering can generate sophisticated whole-system models, and enforce robust standards for verification and validation.

We believe this process is the best way to ensure accuracy and quality in any development process.