# Editorial

**What does good look like?** Early in my career, that question filled me with frustration, especially as it typically arose in 'career progression' interviews. Another favourite, for the questioner rather than for me, used to be: Where do you see yourself in five years' time? Much of my frustration arose because I never felt I had good answers to either of these questions. I particularly remember answering the latter one with, 'not dead, yet?', complete with the question mark, as if my employer somehow held the gift of life and death over me!

From a personal development perspective, I still don't have a good answer to the 'What does good look like?' question, although 'not dead, yet' remains a reasonable aspiration. I have, however, become much happier with answering the question in other contexts. Much of the reason for that increased happiness can be traced to something that happened at the Snowbird ski resort in the Wasatch mountains of Utah. There, in February 2001, representatives came together who were 'sympathetic to the need for an alternative to documentation driven, heavyweight software development processes' (see agilemanifesto.org/history.html). At this meeting, the Manifesto for Agile Software Development was created.

This manifesto places significant value on four things: individuals and interactions; working software; customer collaboration; and responding to change. At least in my mind, all of these can be related to the question of 'What does good look like?' or, if you prefer, the 'definition of done', but in my experience the latter two are especially powerful. They move from a situation in which the customer was expected to precisely define a (potentially large) system to one where the customer and supplier can iteratively develop a shared understanding of 'good'.

It's an over-simplification, but I find Figure 1 helpful in this regard. It illustrates how the triumvirate of time, cost and quality are controlled in the pre-agile and post-agile approaches.

In the pre-agile setting, quality is fixed, for example, being specified in a large set of customer-defined requirements: time and cost are, in some sense, variable, being whatever is required to deliver these requirements.

In the post-agile setting, time and cost are fixed (typically, by the number of people in the team and by the duration of a sprint) and quality (i.e. how much of the requirements backlog is addressed) is the variable outcome.

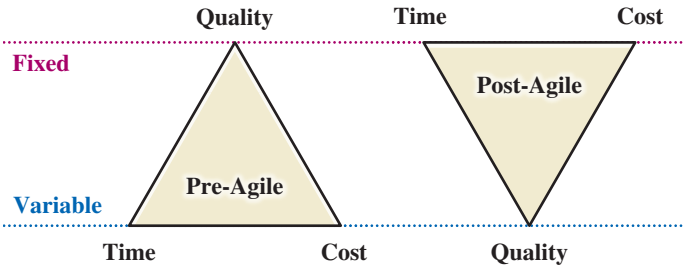


*Figure 1: Pre-agile vs post-agile approaches*

Seen in this manner, which I again emphasise is an over-simplification, the agile approach can be viewed, at least in part, as a reaction to two factors that had historically confounded software development: firstly, it is difficult to specify requirements; secondly, given a set of requirements, it is difficult to estimate the time and cost associated with their delivery.

In saying this, I don't mean to suggest that all software projects are unmitigated disasters, or that there are no good software estimating tools. Conversely, it's just too easy to find examples of important software projects that did not go to plan. More generally, the adoption of agile-like approaches in other disciplines suggests that requirements and estimation are not unique challenges for software.

Specifying requirements is difficult. It's difficult because the English language is imprecise. That imprecision is great for cryptic crosswords and humorous jokes, but it's not helpful when trying to communicate detailed ideas between communities with different specialisms and different vocabularies. There are formally defined languages that can provide precision but, at least in my experience, too few people are fluent enough for these languages to be of widespread practical use.

> … One challenge with uncertain futures can be that all options look similarly good …

Another reason specifying requirements is difficult is because they are affected by changes in the external environment in which the software operates. For large-scale systems that can take several years to implement, I'd argue that these types of change are almost inevitable.

From a mathematical perspective, we can view this as an optimisation problem. We're trying to find inputs (i.e. to write requirements) that optimise an objective function (i.e. system behaviour) in a situation where that objective function alters over time. We may also be in the situation where, even for a fixed objective function, there is some uncertainty in the function's value.

Over recent years there has, I think, been significant progress in addressing uncertainty. For example, there's been a growing (and, in my view, helpful) trend to consider uncertainty quantification (UQ). The Dakota tool, from Sandia National Laboratories, is a powerful example of this (dakota.sandia.gov).

Obviously, I don't see everything, and this isn't an area I've researched in detail, but I'm not sure we have powerful enough tools for addressing changes over time. Equivalently, I'm not sure we have good ways for valuing future agility or, perhaps, for helping people perform decision-making under uncertainty.

There are, of course, tools within the financial sector, like the Black–Scholes model, that can be used to represent future value in a rigorous and auditable manner. Despite their clear utility, I'd argue that such tools operate in a pretty simple environment. For example, if we're just concerned with profit or loss then the objective function is one-dimensional, so future changes cannot encompass situations where there are significant changes in the relative importance of different dimensions within a multi-dimensional objective function.

Even if we had tools, I wonder whether they would be able to offer meaningful guidance. One challenge with uncertain futures can be that all options look similarly good, or similarly

bad, because averaging over lots of potential futures can drag everything towards the mean.

Another challenge is over-specialisation, where we have options that are brilliant in some situations but terrible in others. If we could predict which future situation will occur then this would be useful information, but often we cannot do that.

Likewise, the information would be valuable if we could afford to implement multiple options (equivalently, in financial terms, if we could take a portfolio-based approach), but often that's not possible either.

Overall, I think this means the question we began with has now morphed. It's morphed into something that mathematics ought to be able to help us with. But, it's also morphed into something I find even more challenging to answer, specifically:

> What does good look like when answering the question, what does good look like?

**Rob Ashmore CMath CSci FIMA**
Defence Science and Technology Laboratory

*Artist: Ron Weickart, www.network-graphics.com*