Requirements

Requirements can be a little problematic. User, system and technical requirements are difficult enough to define well in a customer-driven field where there is a defined customer who is expert and knows exactly what they want, but in a field with a non-expert customer or no customer, they can be close to impossible.

In innovations, there is typically no defining customer, so you have to create requirements against a series of expert guesses about who the end user is, what that end user is likely to want and to use the system for, who the buyer is (and their own estimate of the end user, which is itself likely to differ from the eventual end use) and against market moves that you can only roughly predict from what you can see of your potential competitors, suppliers and market-makers (where they exist too).

And if you're innovating in a fast-moving field (e.g. consumer electronics or finance), the merit of carefully creating and working against requirements can itself be debatable because the field could itself move on whilst the requirements are being created. At which point we get into spiral development, rapid application development and other agile requirement-development models. Most of which, from the outside at least, look rather chaotic (in the bounded rational sort of sense of chaos). Just ask yourself: how many projects has Google delivered on time recently? And how much does that matter?