Start with a Minimally Viable Team

Imagine for a moment you want to build a new product, with a new team, and a new project.

Of course you want to do this Agile. The team will be self-managing and largely autonomous. The initial product will certainly be an minimally viable product.

That might be an MVP in the sense of "a small product which probes the market, will add to our understanding and likely fail." Or it might be an MVP which is "the smallest set of features which will provide a fully functional product." Right now it doesn't matter.

Now the question: *where do you begin?*

How many people need to be on the team? What skills? How long should the project last? And above all else: how much will it cost?

I know of one large European bank which averages 27 months from idea to production. Another company I met were yet to begin work on a project they had been talking about and planning for five years.

All that time getting ready to build something, perhaps that explains why so many projects are late?

Lets consider the traditional, rational, way to start work. Then let me suggest a better way: the Minimally Viable Team.

A rational project start

First ask an analyst to consider what you need to address the opportunity. The analyst probably comes up with a list of features, hopefully they come up with a statement of the expected benefits, customers and stakeholders.

As well as the analyst you need an architect to recommend technology, software design and staffing. Do you need a big team or a small one? Do you need a database specialist? A user experience designer? How many programmers? Testers?

Pulling this altogether into a coherent business case requires a project manager. Adding a PM to the squad brings financial discipline and forces consideration of cost.

Now, you have your proposal. This can go before a review body who might grant funding, or might knock it back. (And if it gets knocked back the trio might revise it and try again.)

Finally, after weeks of pre-project you are ready to begin. You start to hire the team, if you are very lucky one or two of the pre-project team might join. Over the next few weeks team members start to appear. And again the Forming-Storming-Norming and Performing pattern plays out, only at the end do you have a productive team.

But...

That autonomous self-managing team aren't really that autonomous. The Analysts has bequeathed them a requirements document and/or an end state. They team only have limited ability to change either of those.

The architect bequeathed them a design to follow. In choosing who to hire, and what skills, the architect has determined much of the design and architecture.

Similarly the business case and plan produced by the project manager and agreed by the review committee has similarly constrained the team.

On the one hand, this is good governance. The corporation has constrained what the team can do.

On the other hand, this team is far from autonomous. If they were to suddenly find a short-cut to the end-state using new technology there are few incentives for them to take it. Indeed, if the new technology only needs half the team, and half the budget, there are good reasons for sticking with the original plan.

Deviating from the business and technology plans given to the team carries personal risk. Subsequent problems could be seen as stemming from the decision to deviate from the plan. Sticking with the plan is safer for team members because others can be blamed for problem.

Doing the pre-project work, recruitment and team formation all took time. The opportunity has been changing all this time. Competitors might also have begun work, or even entered the market.

Delay in starting a work becomes delay in delivery. And cost-of-delay means delayed delivery means lost value.

The Minimally Viable Team alternative

With an MVT you pull together a tiny team as soon as you can justify a little seed funding. That is probably two or three people. The team begins work immediately. Small budgets mean little risk and tight feedback cycles.

The team will analyse the problem, the opportunity and potential solutions. But they will also begin building solutions at the earliest opportunity.

Initial team staffing includes analysis and development skills. But rather than spend time considering what those skills might be - which requires a design, which requires a problem statement, etc. - initial staffing is approximate and most likely includes an experienced analyst and an experienced coder.

The team can do traditional analysis if they want but they should also use prototypes, proof of concepts and models they can show potential customers. The team learn from analysis, planning and building, more importantly they will kick-start the feedback cycle and allow the team to market test ideas far sooner.

A small team will form-storm and norm much more quickly than a large one and reach performing far sooner. If things look good the team will pull in more skills and people. The team will need to demonstrate a successful track record of both product discovery and product development.

Governance looks more like venture capital than traditional corporate portfolio management. The MVT has a little money and a deadline: a month, three months, whatever the money runs out. At the end of that time they must show what they have learned, built and propose the next move.

If thing look positive they get granted more time and more money, possibly enough money to grow the team. If not, then its game over and everyone moves onto something else.

Working like this gives the team more autonomy. It also avoids sunk costs: at each review point the team show what they have learned and built.

By effectively back-dating the project to point where pre-project work normally happens opens options and improves governance.

Too big to fail

Traditional projects have a habit of becoming too big to fail. Large projects, with large sunk costs, continue to attract more money because failing to fund failing work raises questions about how money was initially allocated. Such questions become threatening to those who initially allocated the money thereby creating an incentive to throw good money after bad in the hope that everything will work out.

The MVT approach prevents too big to fail because each piece of work is small, no one team or project will threaten a career. Indeed, teams are almost expected to fail because they are working in risky fields.

MVTs have one more benefit: because teams get starved of resources they cannot afford to do more than the minimum. Like a start-up every penny must count. And because teams have minimal staffing nobody needs to make work to justify their role.

Or to put it another way: if you want to build an Minimally Viable Product use a Minimally Viable Team.