Solving difficult problems with lean thinking

— Dr Nick Scott

“*Everything is hard before it is easy*” — Goethe

The track record of lean organisations with problem solving is impressive. This is how they move forward, both in continuous improvement of their current work and breakthroughs into new markets and technologies. We believe that borrowing their overall approach to problem solving, mastering the underpinning theory and using the detailed practices, provides new insights and capability.

The observations in this paper are based upon the author’s experience helping many client organisations to adopt lean thinking in order to get better at solving difficult problems in their business, technology, in daily operations and use of strategy.

Think System

Lean thinking encourages us to visualise the problem we are trying to solve as the outcome or state from a system. We can then define the problem system as one which currently produces an undesirable outcome.

Furthermore, recalling what we said in the first article in this series about measurement, we can define our problem (outcome) in a quantitative way. This may seem obvious, but it is the first step in unravelling the perceived complications that makes a certain class of problem so difficult.

Put another way, if we quantify what’s *bad*, we can put a target (in lean we call this our target condition) for what *better* will look and feel like. What’s more, we can agree what *good* enough will feel like. This is not simply a vague setting out of goals. A problem only becomes real once we can measure it.

Lean problem solving can therefore be seen as the process of defining problems in such a way that we can identify the causes of the problem states we wish to change. If we can measure the causes, we can create leading measures for the problem itself. If we understand the causes rigorously enough, going back to a root cause we can act upon, then we have real guidance for a workable solution.

What is a root cause?

A root cause is defined as the fundamental or initiating cause of an outcome from a system. We focus on addressing root causes because we prevent the recurrence of the problem *by that means*.

It is surprising how often in our lives we treat problem symptoms and ignore root causes as either too hard to master or we don’t think it can form the basis of concrete action. As intuitive as root causes appear to engineers, broader society often overlooks them.

We have Joseph Juran, quality guru and management consultant, to thank for his work on root cause. Getting to a root cause can be difficult, but it’s worth the investment.

- We need to know *where* to start our analysis from, so we need to be rigorous in our definition and analysis of the nature of the problem.

- We dig down to root cause by understanding cause-effect relationships in the system we are studying. How do we do this? We keep asking *why*.
- Root causes are fact based, apolitical, simple and *don't tolerate blame culture*.
- Mastering root causes improves our knowledge and presents us with *learning opportunities*.

So what makes a problem difficult?

We can solve most problems if we understand them well enough. There is, however, a class of problems called Wicked Problems, where interdependencies between system components make for difficult resolution. To be more specific, the interdependency stems from interactions that are difficult to predict because:

- We do not understand all of the causes of the behaviour we are seeing;
- Small changes produce large, unexpected shifts, popularised as *tipping points*;
- The system is sensitive to its initial conditions. A slight change in starting point takes the system down a different route to different outcomes.

A system which is known to exhibit one or more of these properties is more risky to simplify or reduce into its parts for separate solution. In these systems, well intended, seemingly robust solutions don't give expected results when widely applied, create unintended further problems and occasionally *fall off a cliff* with catastrophic and unexpected results.

But it's not all bad news.

Mercifully few problems are genuinely wicked. The international market for mortgage bonds may be complex and, as we saw in 2008, it became unstable. Lots of us have mortgages but that doesn't mean our *individual mortgage* lives will be so unstable.

It takes the agency of many individuals and other macroeconomic systems to be out of control — in very predictable ways as it turned out — for this instability to emerge at a higher level of abstraction. So, the trick is to set our system view at the right level(s) of abstraction.

How do we choose? We set it at the level most acutely perceived by our customer *and* the agents in the system who cause changes. This may not be the same level. Part of our new agility in visualising the problem we are solving is getting used to checking between levels.

It is also important to understand the historical baggage we are carrying that makes the problem more difficult to understand. Is all the information in one place and expressed in terms we can all understand? To what extent have our norms, systems and procedures built up and become part of the perceived difficulty of the problem? If we take time to strip away some of that *complication*, how difficult will the problem appear?

In lean, going back to basics to see clearly is a near-obsession. This is underpinned by our focus on Gemba (Japanese for *real place*, ie where the customer's work is performed).

How we describe and define the problem will also have a huge impact on its actual reducibility.

What aspects of the system's complexity must/should we accept? What can we challenge? Are we trying to transform a rich and diverse system to perfection, once and for all? Would we settle for a flow of smaller, easier problems to work on with pace and enthusiasm if we could be sure we are really getting to grips with the problems? How do we feel about revisiting a problem multiple times to nudge it along? Is this wasted effort or just a different mindset at work? Is this a fertile training ground for expert problem solvers or busy fools? How do we ensure the former and prevent the latter?

Hardly game-changing, is it?

When first working with lean thinking to make things better, the problems are not difficult and the solutions often yield surprisingly large results. When first exposed to lean — and the three clues to problem solving — lots of us initially miss the point.

For example, say we wanted to reduce the time it takes a technician to change a machine over from making *Product A* to *Product B.* One of the first things we may do is move the tools closer to the technician. Seems pretty obvious — but are we really going to leapfrog the competition by doing things like this?

But wait. Rewinding what we have just done, we defined the problem a particular way, identified a root cause of the current state we could act on easily, implemented something relatively quickly to test, and we involved the technician in the change.

Next, we create the conditions that allow the technician to keep working on the problem, on the condition that they work towards the same goal, test their ideas frequently, and check they are not creating another problem somewhere else.

Keep thinking and acting like this and you may just be changing the game after all. This example, Single Minute Exchange of Dies (SMED) in Toyota's production system, is a key competitive weapon.

Let's take a look at a worked example of a design problem.

We acknowledge in all of this that we are talking about a particular class of problem. A good example of this is the realisation the operations management community had when confronted with cellular manufacturing in the 1980s and 1990s. This was actually a design problem extraordinaire. We were faced with the hard computational problem of optimising the output from our large functionally organised factories (all machines of one type located in groups together to ensure they were loaded up to peak efficiency – *ie always busy*).

In reality, efficiency from these functions meant that we had queues of work in front of these groups. This gave us the unintended consequences:

- Queues build waiting time, so to meet precise customer demand we had to forecast (aka *guess*) what to make. The result: we were probably not making what customers wanted at any one point in time.
- Queues increase the number of parts to be scheduled, increasing the computational problem.
- The increased number of interdependent parts in the system make small changes to individual operations more likely to produce large changes in overall system behaviour. It becomes more difficult to produce a schedule we can trust.

The problem was a spiralling arms race between increasing size and complexity of the system and our ability to calculate schedules (predictions) for how the system might behave. The only way to break the system appeared to be investment in new, faster, automated machines. To ensure this investment worked, these machines had to be loaded *even more efficiently*. One way to pay for this was to reduce the number of people needed in the factory.

Lean problem solving defined the problem differently.

Instead of overall system efficiency, we reduce each customer's wait time whilst by trying to minimise idle labour time. On that basis, we would ideally create a factory for every instance of customer demand without queues.

The practical implementation of this is to create *cells* or mini factories by grouping products with comparable process requirements, splitting up functionally similar machines, and allocating them to product *cells*. This effectively creates factories focussed on each of our product types. This focus on the products our customers want eliminates the scheduling problem and creates a new simplicity.

But wait. Doesn't this trash our efficiency? Don't we need to ensure that machines are always busy?

Well, if we abandon that requirement on the system, and replace it with the requirement to keep the people working in the *cells* occupied in useful work, the business does not perform worse. Customers *prefer* short lead times. Even if they don't want products sooner, they like that it helps us keep our promises to them.

How can we justify this wholesale shift in philosophy?

We have broken the vicious circle between efficiency and investment. We don't burden the business with investment costs because our focussed factories use simpler, slower machines. We grow our expertise in-house to create our own right-sized equipment, what professor of operations management John Bicheno calls his Theory of Small Machines. We also ensure that each operator learns how to run multiple machines in each cell, a practice not possible when machines cluster by functional group. Labour efficiency increases, people learn more and the organisation builds process knowledge — instead of buying it.

However, we do impose constraints on our problem solving. In this case, we don't allow any more factory space or people to create our cells — we have to work within floorspace and headcount.

Let's now leave the high-level approach, and discuss some of the rich detail of how to do lean problem solving.

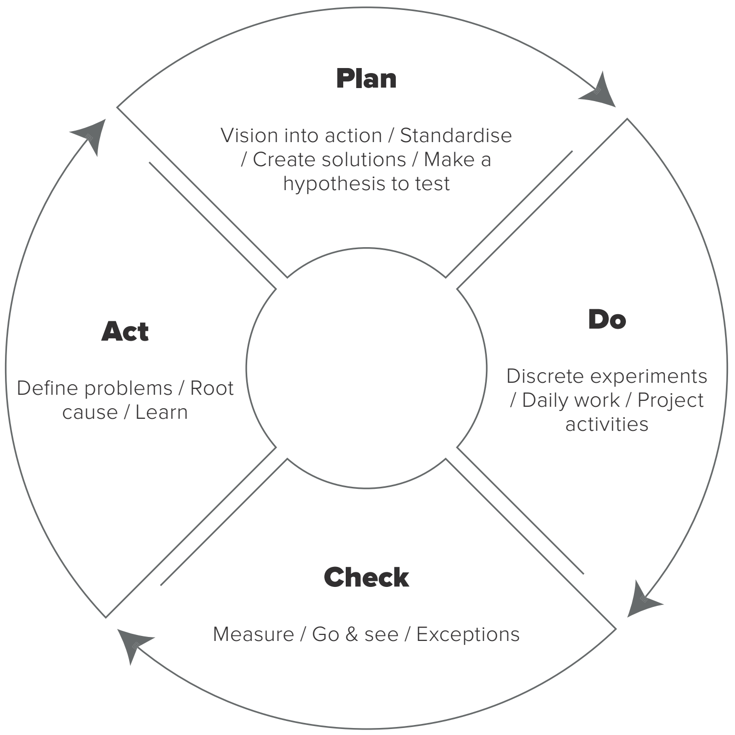
You know what they say about a good plan...

With our focus on root causes, we base our problem-solving choices on facts. Armed with this, creating a solution is an opportunity to run experiments.

The lean way to perform an experiment is a Plan-Do-Check-Act (PDCA) cycle, for which we thank W Edwards Deming, statistician and teacher.

The PDCA sets out a Plan — our hypothesis to test — we then Do (the test, which can be a formal set-piece experiment or just our daily work), and we Check (against the definition of success – ie measure - set out in our Plan). Finally, we Act on what we have seen by reflecting on the hypothesis, improve our understanding of actual root causes, and we learn.

And then we Plan again ... (that's why we call it continuous improvement!).


Plan
Vision into action / Standardise / Create solutions / Make a hypothesis to test
Do
Discrete experiments / Daily work / Project activities
Check
Measure / Go & see / Exceptions
Act
Define problems / Root cause / Learn

There are a couple of important things to remember about PDCA cycles:

1. Everything we do in a lean thinking environment is a PDCA cycle, even our daily work. So whilst we might think about problem solving as a discrete event, it is taking place constantly on all scales and clock-speeds within the organisation. By making this explicit, you can't help but learn from your experiences.

   How would your job feel if you were able to view its complexity as a series of problem solving cycles?

2. Sticking to the discipline of a PDCA cycle addresses the phases of good problem solving: being really clear what problem we are solving and being able to measure *better* (or *worse*). We are interested in investigating root cause and the solutions that arise from this investigation, not simply off-setting the symptoms we are experiencing. The solution needs to be tried, evaluated and only left in place once we are convinced that it really addresses the problem (as defined by our stated measure of success).

How does this work in practice with lean thinkers?

Lean problem solving tells us not to wait too long once a problem arises. In lean thinking environments we create windows of opportunity to stop and problem solve.

Sometimes our cycle starts with Check. This is called a CAP-Do cycle, but it is just a PDCA with an emphasis that suits its context, such as monitoring daily work.

We wish to control in short intervals, even if *every day is going to be different* such as an R&D environment. How can we do this? Well, most work — from the repetitious to the novel — is driven by a Plan.

The other benefit of not letting the grass grow under our feet is that we minimise the number of problems we have to deal with at any one time, increasing the likelihood that we can bring enough team focus to bear on the problem to solve it quickly.

We must also *protect* the customer.

In practice, this means — despite what we have said about symptoms — putting something in place to prevent the people affected by the problem from suffering its ill effects.

This can have nothing to do with the root cause. The rule is that we do this but only with a view to removing it once we have verified (following PDCA) that our root cause solution is working. Then the temporary fix has got to go. How often do we stop and move on once an initial fix has been put in place to contain the problem? This only builds cost and complexity into our system.

We use the insights of root cause to help us find simple (or even *obvious*) solutions. The conventional wisdom in problem solving can be to *brainstorm* to map out a huge space of potential solutions. This is, of course, sometimes completely appropriate. But if we know (or think we do, pending our PDCA cycle!) a root cause, it is often quite obvious and straightforward how to influence it for the better.

Sometimes it is about *not overthinking the solution* to a well-defined root cause. Just get on and test it!

And remember that we are all in it together.

Problem solving is also a *team sport* and we must *create the conditions* for lean problem solving. Colleagues and bosses are here to help as best they can. To help them help us, the problem has to be transparent and a matter of public record within the team.

To do this, we manage problems visually and these are often displayed in the workplace, using the same terms of reference that enable lots of people to understand the *work at a glance*. Examples of this include A3 problem solving, so-called because the problem description, analysis and experiments in solving it are succinctly presented on A3 sized sheets on display in the team's workplace.

The class of problem we are really referring to is the *looks complicated, looks expensive, looks risky and lots of people will have an opinion* type that present themselves constantly in our work and beyond.

Summing up

We have tried to show in this paper that in order to get good at solving these problems, we need to be disciplined in using what are essentially some very simple steps from lean thinking. We also need to ensure our teams get lots of practice, and that over time we build knowledge of our problems and their solutions.

This means better solutions to problems, building the depth of understanding of the systems in which problems occur, and building our teams into expert problem solvers.

Thank you for taking the time to come along on this exploration of lean problem solving. We hope that we have been able to demonstrate how the application of a few principles of lean thinking can offer a fresh route to solving difficult problems.

In our next article, we will put this all together and talk about what lean systems can teach us about constructing improved product creation processes.

Until then, happy problem solving!

Dr Nick Scott
Senior Consultant

Nick is an industry experienced manufacturing engineer specialising in the application of lean thinking to the improvement and creation of operational processes and products.

His experience includes inkjet printing, speciality chemicals, functional packaging, medical devices, precision gauging and inspection equipment and capital machinery. He has also applied these ideas to healthcare delivery, utilities replacement and retail banking.

Nick has an honours degree in Manufacturing Engineering from the University of Cambridge and holds a PhD in complex systems within operations management. He is a chartered engineer and has also been an operations and value stream leader.