The law of requisite variety and its implications for enterprise IT

K

11 months ago

Introduction

There are two facets to the operation of IT systems and processes in organisations: *governance*, the standards and regulations associated with a system or process; and *execution*, which relates to *steering* the actual work of the system or process in specific situations.

An example might help clarify the difference:

The purpose of project management is to keep projects on track. There are two aspects to this: one pertaining to the project management office (PMO) which is responsible for standards and regulations associated *with managing projects in general*, and the other relating to the day-to-day work of *steering a particular project*. The two sometimes work at cross-purposes. For example, successful project managers know that much of their work is about navigate their projects through the potentially treacherous terrain of their organisations, an activity that sometimes necessitates working around, or even breaking, rules set by the PMO.

Governance and *steering* share a common etymological root: the word *kybernetes*, which means *steersman* in Greek. It also happens to be the root word of *Cybernetics* which is the *science of regulation* or *control*. In this post, I apply a key principle of cybernetics to a couple of areas of enterprise IT.

Cybernetic systems

An oft quoted example of a cybernetic system is a thermostat, a device that regulates temperature based on inputs from the environment. Most cybernetic systems are way more complicated than a thermostat. Indeed, some argue that the Earth is a huge cybernetic system. A smaller scale example is a system consisting of a car + driver wherein a driver responds to changes in the environment thereby controlling the motion of the car.

Cybernetic systems vary widely not just in size, but also in complexity. A thermostat is concerned only the ambient temperature whereas the driver in a car has to worry about a lot more (e.g. the weather, traffic, the condition of the road, kids squabbling in the back-seat etc.). In general, the more complex the system and its processes, the larger the number of variables that are associated with it. Put another way, complex systems must be able to deal with a greater *variety* of disturbances than simple systems.

The law of requisite variety

It turns out there is a fundamental principle – the law of requisite variety– that governs the capacity of a system to respond to changes in its environment. The law is a quantitative statement about the *different types of responses* that a system needs to have in order to deal with the range of disturbances it might experience.

According to this paper, the law of requisite variety asserts that:

> The larger the variety of actions available to a control system, the larger the variety of perturbations it is able to compensate.

Mathematically:

$$\mathbf{V(E) \geq V(D) - V(R) - K}$$

Where **V** represents variety, **E** represents the essential variable(s) to be controlled, **D** represents the disturbance, **R** the regulation and **K** the passive capacity of the system to absorb shocks. The terms are explained in brief below:

V(E) represents the *set of desired outcomes for the controlled environmental variable:* desired temperature range in the case of the thermostat, successful outcomes (i.e. projects delivered on time and within budget) in the case of a project management office.

V(D) represents the *variety of disturbances the system can be subjected to* (the ways in which the temperature can change, the external and internal forces on a project)

V(R) represents the *various ways in which a disturbance can be regulated* (the regulator in a thermostat, the project tracking and corrective mechanisms

prescribed by the PMO)

K represents the *buffering capacity of the system* – i.e. *stored* capacity to deal with unexpected disturbances.

I won't say any more about the law of requisite variety as it would take me to far afield; the interested and technically minded reader is referred to the link above or this paper for more.

Implications for enterprise IT

In plain English, the law of requisite variety states that only "*variety can absorb variety.*" As stated by Anthony Hodgson in an essay in this book, the law of requisite variety:

> ...leads to the somewhat counterintuitive observation that the regulator must have a *sufficiently large variety of actions in order to ensure a sufficiently small variety of outcomes in the essential variables E.* This principle has important implications for practical situations: since the variety of perturbations a system can potentially be confronted with is unlimited, we should always try maximize its internal variety (or diversity), so as to be optimally prepared for any foreseeable or unforeseeable contingency.

This is entirely consistent with our intuitive expectation that the best way to deal with the unexpected is to have a range of tools and approaches at ones disposal.

In the remainder of this piece, I'll focus on the implications of the law for an issue that is high on the list of many corporate IT departments: *the standardization of IT systems and/or processes.*

The main rationale behind standardizing an IT process is to handle all possible demands (or use cases) via *a small number of predefined responses.* When put this way, the connection to the law of requisite variety is clear: a request made upon a function such as a service desk or project management office (PMO) is a *disturbance* and the way they regulate or respond to it determines the *outcome*.

Requisite variety and the service desk

A service desk is a good example of a system that can be standardized. Although users may initially complain about having to log a ticket instead of calling Nathan directly, in time they get used to it, and may even start to see the benefits... particularly when Nathan goes on vacation.

The law of requisite variety tells us successful standardization requires that all possible demands made on the system be *known* and regulated by the **V(R)** term in the equation above. In case of a service desk this is dealt with by a hierarchy of support levels. 1st level support deals with routine calls (incidents and service requests in ITIL terminology) such as system access and simple troubleshooting. Calls that cannot be handled by this tier are escalated to the 2nd and 3rd levels as needed. The assumption here is that, between them, the three support tiers should be able to handle majority of calls.

Slack (the **K** term) relates to unexploited capacity. Although needed in order to deal with unexpected surges in demand, slack is expensive to carry when one doesn't need it. Given this, it makes sense to incorporate such scenarios into the repertoire of the standard system responses (i.e the V(R) term) whenever possible. One way to do this is to anticipate surges in demand and hire temporary staff to handle them. Another way is to deal with infrequent scenarios outside the system- i.e. deem them out of scope for the service desk.

Service desk standardization is thus relatively straightforward to achieve provided:

- The kinds of calls that come in are largely *predictable.*
- The work can be *routinized*.
- All non-routine work – such as an application enhancement request or a demand for a new system- is dealt with *outside* the system via (say) a change management process.

All this will be quite unsurprising and obvious to folks working in corporate IT. Now let's see what happens when we apply the law to a more complex system.

Requisite variety and the PMO

Many corporate IT leaders see the establishment of a PMO as a way to control costs and increase efficiency of project planning and execution. PMOs attempt to do this by putting in place governance mechanisms. The underlying cause-effect

assumption is that if appropriate rules and regulations are put in place, project execution will necessarily improve. Although this sounds reasonable, it often does not work in practice: according to this article, a significant fraction of PMOs fail to deliver on the promise of improved project performance. Consider the following points quoted directly from the article:

- "50% of project management offices close within 3 years (Association for Project Mgmt)"
- "Since 2008, the correlated PMO implementation failure rate is over 50% (Gartner Project Manager 2014)"
- "Only a third of all projects were successfully completed on time and on budget over the past year (Standish Group's CHAOS report)"
- "68% of stakeholders perceive their PMOs to be bureaucratic (2013 Gartner PPM Summit)"
- "Only 40% of projects met schedule, budget and quality goals (IBM Change Management Survey of 1500 execs)"

The article goes on to point out that the *main reason for the statistics above is that there is a gap between what a PMO does and what the business expects it to do*. For example, according to the Gartner review quoted in the article over 60% of the stakeholders surveyed believe their PMOs are overly bureaucratic. I can't vouch for the veracity of the numbers here as I cannot find the original paper. Nevertheless, anecdotal evidence (via various articles and informal conversations) suggests that a significant number of PMOs fail.

There is a curious contradiction between the case of the service desk and that of the PMO. In the former, process and methodology seem to work whereas in the latter they don't.

Why?

The answer, as you might suspect, has to do with variety. Projects and service requests are very different beasts. Among other things, they differ in:

- **Duration**: A project typically goes over many months whereas a service request has a lifetime of days,
- **Technical complexity**: A project involves many (initially ill-defined) technical tasks that have to be coordinated and whose outputs have to be integrated. A

service request typically consists one (or a small number) of well-defined tasks.
- **Social complexity**: A project involves many stakeholder groups, with diverse interests and opinions. A service request typically involves considerably fewer stakeholders, with limited conflicts of opinions/interests.

It is not hard to see that these differences increase variety in projects compared to service requests. The reason that standardization (usually) works for service desks but (often) fails for PMOs is that the *PMOs are subjected a greater variety of disturbances than service desks*.

The key point is that the increased variety in the case of the PMO precludes standardisation. As the law of requisite variety tells us, there are two ways to deal with variety: regulate it or adapt to it. Most PMOs take the regulation route, leading to over-regulation and outcomes that are less than satisfactory. This is exactly what is reflected in the complaint about PMOs being overly bureaucratic. The solution simple and obvious solution is for PMOs to be more *flexible*– specifically, they must be able to adapt to the ever changing demands made upon them by their organisations' projects. In terms of the law of requisite variety, PMOs need to have the capacity to change the system response, V(R), on the fly. In practice this means recognising the uniqueness of requests by avoiding reflex, cookie cutter responses that characterise bureaucratic PMOs.

Wrapping up

The law of requisite variety is a general principle that applies to any regulated system. In this post I applied the law to two areas of enterprise IT – service management and project governance – and discussed why standardization works well for the former but less satisfactorily for the latter. Indeed, in view of the considerable differences in the duration and complexity of service requests and projects, it is unreasonable to expect that standardization will work well for both. The key takeaway from this piece is therefore a simple one: those who design IT functions should pay attention to the *variety* that the functions will have to cope with, and bear in mind that standardization works well only if variety is known and limited.

Categories: Best Practice, Corporate IT, Enterprise Architecture, IT Management, Organizations, Project Management

Leave a Comment