Designing A Business Immune System

Mr Business? Time for your shot!

As I've been talking a bit about an immune system for businesses at various venues, I'm often asked to describe my view of what such an immune system might be. Towards that end, I thought it would be convenient to gather my thoughts in a series of blog posts. In this first installment, I will describe what it might be on a more conceptual level, by comparing it with a biological immune system. In future installments I will explore ways to implement an immune system and where we can find inspiration for the various parts it may be comprised of.

Most people understand the concept of an immune system as a complex system that work to protect the health and proper functioning of its host organism. We know that without an immune system most biological structures would soon die from disease or other disorders, but even so we accept that some of the structures we humans create and cherish the most go

without the benefit of such a defensive mechanism, in particular our business organisations.

Yet our businesses are under constant attack from various pathogens in many forms; some are external in the form of competition, regulations, changes in the market, poor user reviews and many more; some are internal in the form of poor decisions, discontent amongst staff, wasteful practices, inability to react to changes in the market or users preferences and much more. Any one of which may ultimately lead to the complete failure of our venture. So can we equip our business with an immune system that, to the extent possible, protects against these pathogens?

It’s not as far fetched as it may seem at first, and we’re all familiar with some of the things that could be part an immune system, such as testing software before it goes into production, gathering user input to determine what products to develop, implementing feedback loops from production systems in the form of health monitoring and log handling, and many more. Some of these defences are technical in nature, others are social in nature, but they can all be said to the part of a rudimentary immune system.

Without presuming to have any medical knowledge, I will draw a parallel between the business immune system and the human immune system, since the pattern to which the human immune system is modeled seems to be effective enough to have preserved the human race so far. So what are some of the traits of an immune system?

- It is autonomous
- It is always present
- It has different mechanisms that complements one another, reacting to different signals
- It is self learning and evolving
- Although less desirable, it must be maintained

Let’s examine each of these point, to see how they might apply to an immune system for businesses:

Autonomy

Free to act!

An immune system acts on its own. Where it is possible to automate, this can simply mean that some signals, or class of signals, result in corrective action of some form. An example of a semi-automatic nature might be a programmer that immediately corrects code as a result of a failing test. An example of a fully automatic action could be a monitoring system that scales up or down a cluster, or a Continuous Integration server that rolls back a code commit as a result of a failing test or deployment, possibly after first blocking further code commits for a period of time, to force an immediate correction of the faulty code or deployment script.

Ideally, on a higher level, autonomy means that whoever is performing a certain task also have the authority to perform corrective action as a result of signals or symptoms in the immune system. For instance, the persons performing an experiment to validate a business hypothesis are allowed to pivot or preserve without further involvement from management or governing body. In this case it is important that the persons responsible for the hypotheses under test, are also taking part in performing the validations.

Always present

Always there for you

An immune system never takes a break, and is present at every stage.

In its most general form this means that it is necessary to put in place immune system signals for everything from hypotheses, coding standards, tests, performance, integration points, response times, correct functionality and user behavior. Even an IDE can be a part of this, highlighting various irregularities in the code.

It also means that we never disable any part of the immune system for some imagined quick gain, such as quickly getting a feature into customers hands. By letting the full immune system work for us, the amount of waste and re-work such as working on the wrong thing, fixing bugs and similar, will be far less than without an immune system, allowing us to delight our customers faster yet safer than we otherwise could.

Different mechanisms

Different defenses

An immune system for businesses is never just tests, or just monitoring, but multiple systems that on their own or in concert can take appropriate actions, such as blocking further code commit as a result of a failing test, or even roll back the latest commit, or critical values in monitoring can start new instances, start throttling traffic or roll back the latest production deployment.

A well crafted immune system can even have difference mechanisms acting in concert or having them affecting each others' responses. An example of this could be a monitoring system that will auto scale at a lower limit for response times as it detects a larger number of new customers, all to ensure new customers get the smoothest possible experience.

Self learning and evolving

Self study

The human immune system is always evolving and learns about new pathogens and how to combat them, without conscious or deliberate involvement from us, with the exception of the teaching we do in the form of vaccinations.

There is research with the ultimate goal of emulating some of this capability in software systems using Machine Learning, and and we will explore some of the more promising implementations in later installments. Current applications of self learning is more about automatically adjusting thresholds for when to scale up clusters, or finding the connection between response times and user activities and thus user experience. Attempts are also being made to auto-generate tests based on detected failures, but such techniques are far

from being mainstream.

It is entirely possible to automate the execution of the experiments that are run in order to validate business hypothesis, but the ability to also automatically and autonomously adjust the tested hypotheses, to pivot in Lean Startup terms, are as of now in the realm of science fiction. Although I would love to be proven wrong about that, it would require an insight into human psychology and behavior on the part of the machines that would perhaps seem a little uncomfortable to some.

Maintaining the immune system

Maintenance needed

As we know, an immune system that is not cared for, or neglected during a longer time, can be rendered ineffective or even actively damaging to the host. The same is true for a business immune system.

Each time a new potential pathogen, such as new code and new functionality, is introduced without equipping the immune system with the tools it needs to detect and combat these new risks, the immune system is weakened, with an increasing risk for disease or even death of the system. These tools can be new or adapted tests or extended monitoring.

External changes, such as changes to the environment or surrounding systems, can also affect the system, and even the immune system itself.

Do *we* have an immune system?

Cool tools

So what about our cool tool or framework, doesn't that constitute an immune system? Certainly management seem to expect it to solve almost all the problems in the business?

Well, it depends on how it used. Taking almost any given testing framework, for instance, it is most commonly used more akin to a thermometer, or perhaps a stethoscope, that can to some degree of precision analyse the current health. In many cases it is more like taking vitamin C or similar, which helps empower part of the immune system. In yet other cases it can indeed be part of an active immune system, if it is part of the tests that are always run on any change and there are appropriate actions it can take when faults are detected.

A little too late?

But we have a testing department, and our operations team is using a cool tool for monitoring (although they are in India, so this is mostly hearsay)?

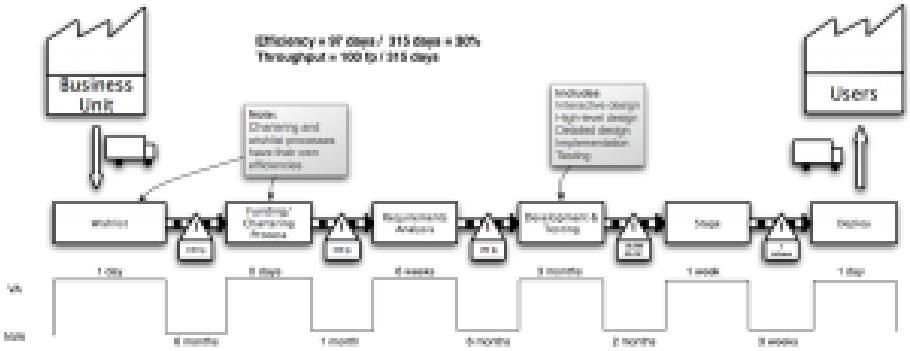
Then you have a lot of nurses, and perhaps even a doctor or two, but not much of an immune system to speak of. Most likely your nursers and doctors are very busy, always stressed and unable to effectively help the patient in a safe and

predictable way.

Introducing Leftshifting

In the last few years there we have seen an ever increasing awareness about more streamlined practices for developing and delivering value through software and IT. At virtually every single conference you can find sessions on topics such as Continuous Delivery, Automatic Testing, Automatic Deployment, DevOps, Docker and Agile, and more often than not we are told that we have to *shift left*. All too often these practices are discussed as having some intrinsic value of their own, and every so often there is a session devoted to the intricacies of how to convince managers or developers, tester and operations that these practices should be introduced. But as for why we would want to introduce it, we are often assumed to somehow "know" that this is the right thing to do, and that it will make things better.

Interestingly, if we instead start by asking ourselves what kind of outcome we want to achieve by introducing these practices we may find that we are engaged in *local optimisation* in Lean speak, seen from a total systems perspective. That is, we optimise small parts but ignore the overall system, or indeed make the overall system worse. Using more words from Lean, we are not doing *Systems Thinking*. And when we start asking ourselves ***why*** we want to achieve these outcomes in the first place, we may very well discover that there are a whole slew of other practices, many unrelated to the technical work, that will have an even larger impact on the outcome.



A typical Value Stream Map.

It beautifully highlights why the introduction of a more effective software development, which seem to be the pet project of so many managers, have such limited effect on the overall capacity of a software organisation. In this example, reducing development and testing to zero by some magical means, would still yield no more than 15% improvement of Time To Market, with the corresponding 15% impact on the Cost of Delay. It is clear that creating a more effective organisation as a whole is needed. Also note that in this very typical process, the queue times amounts to more than 85% of the total process time.

You need feedback to learn

Does this mean that we advocate against these practices? Most emphatically NOT! We believe it is almost impossible to overstate the importance of introducing a high performing IT delivery, as it is by far the strongest enabler of a high performing business in general. One of the reasons it is such a strong enabler lies in the nature of learning; as it turns out, all learning is improved by good feedback, and the faster and more accurate feedback we get, the better we learn. And of course, learning is the basis for all improvements.

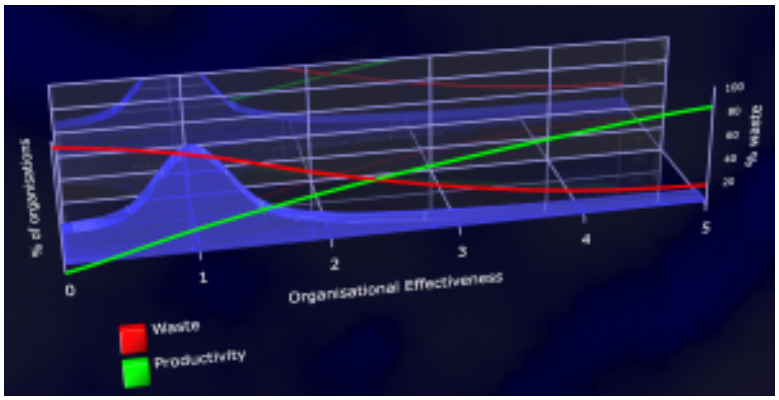
Virtually all of the practices that are now discussed and presented and mulled over so often, have one thing in common; they enable faster and more accurate feedback. Lining up all the activities required to develop a software based product along a time axis, we clearly see that these practices *shift activities left*. This is why you no doubt have heard said that

we have to *shift things left* over and over in the last few years.

So what is the outcome we are looking for? Obviously, there are many different answers to this question, but what we have seen over the years is that most of them can conceptually be boiled down to the same thing; we want our business to become more effective. This inevitably leads to the question of how we define *effective*, and here we have borrowed from Bob Marshall's concept Rightshifting; "effectiveness is a business ability to achieve its goals". As you can see, we were also inspired by the Rightshifting name when we created this space.

Shift left or shift right?

A quick refresher on the Rightshifting concept; Through his work with a huge number of knowledge businesses, together with the quantitive work of Steve McConnell and others, Bob Marshall found that if you were to index businesses' effectiveness on a scale, the resulting curve would not be a classic standard deviation curve that peaks in the middle and have equal distribution to either side. Instead, the curve would be compressed towards the left of the axis with the the vast majority of business between 0.5 and 1.5 on a scale of about 5.



This is the iconic picture of Rightshifting, and the thing to note here is where the vast majority of companies are in terms of waste. Yes, indeed, at 80% waste! 4 workdays out of 5 amounts to no value being added to the company! Since the vast majority of companies are at this stage, you should probably ask yourself, where is your company on this axis?

Value is on this axis created by shifting right, and is at the heart of all the improvements that companies strive for. Since some of the most powerful tools at our disposal that enables a business to *shift right* along the *effectiveness* axis, is to *shift left* along the *time* axis for activities across the whole organisation. We will therefore use this space to investigate, discuss, advice and learn about how we can go about shifting all the activities in our businesses to the left. *Shift left to shift right.*