Why IT struggles to innovate

AND HOW YOU CAN FIX IT

The state of IT

Considering a recent *New York Times* article stating that enterprise-wide, IT spending will be $2.68 trillion in 2013[1], one would think the IT industry – with all its engineering brain power – would be at the forefront of efficiency and automation.

Curiously, this is not the case. If what Gartner reports[2] is true, 80% to 85% of an IT department's budget is used simply in **Keeping the Lights On (KTLO)**.

Let that sink in a moment: At least, $2.68 *TRILLION*[3] in total spending and 80% to 85% of IT budgets used on what is essentially maintenance. The cost of KTLO has never had more zeroes.

Money wasted on maintenance is a sad enough story, but there is a second rail of inefficiency – **endemic to all industries and geographies** – that is even more troubling. It is based entirely on the accumulation of new service or change requests made by business units that IT has a hard time delivering. This failure to deliver is due to limitations ranging from a lack of budget to a lack of resources and skills.

This second rail of inefficiency is usually referred to as an IT department's **backlog**, and is no small matter. It is exactly what causes IT departments to be perceived of as incompetent, slow to deliver and unwilling (or unable) to support innovation.

And in most companies IT backlog is becoming a crippling problem. According to Gartner, IT backlogs are actually **compounding** annually at a rate of 10% to 20% – meaning in the worst cases, companies could see them double in five years!

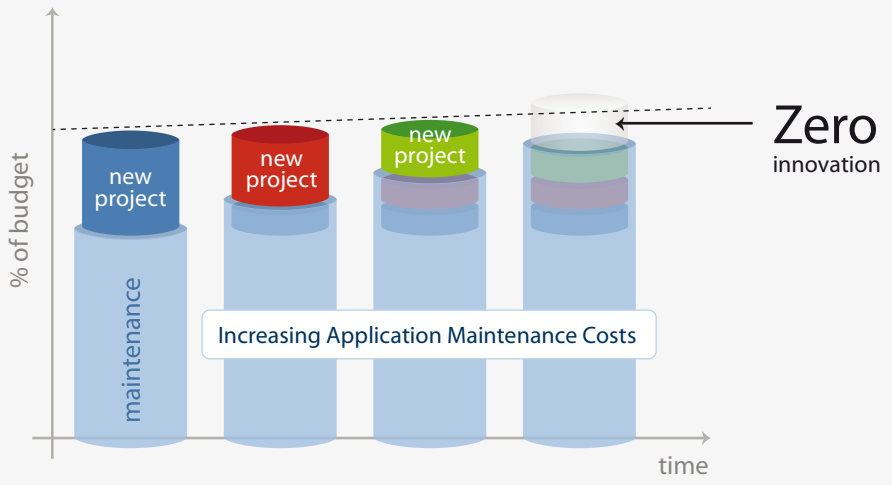
[1] http://bits.blogs.nytimes.com/2012/11/17/hard-times-could-create-a-tech-boom/
[2] http://www.gartner.com/newsroom/id/497088
[3] http://www.gartner.com/newsroom/id/2292815

Business units faced with a critical need for new applications, and an IT department that is unable to respond, will end up doing one of two things:

- Find a way to solve the problem by building their own applications or 'renting' cloud-based solutions – solutions that do not require IT's involvement.
- Hire a systems integrator or boutique consultancy to build the application.

The **Keep The Lights On (KTLO)** Dilemma

Long-term TCO leads to **zero** innovation



If successful, the application will 'stick'. And with continued investment and use the application will eventually need to be turned-over to IT. Now IT has just inherited another application they did not buy or build, and just added more to their KTLO budget and activity.

For most companies it seems like any attempt to fix the inefficiency problem only makes it worse. Technical debt grows. Backlogs get bigger. IT is left behind, unable to react to the needs of business. It is a cycle of spending money, losing ground, and disappointing management.

Given the exorbitant amount of money in play, no organization can close their eyes to so much wasted productivity and spending. So why do so many organizations continue along the same path expecting something different?

The way companies are currently dealing with their backlogs and technical debt is not working. It is an unsustainable business model that will render IT departments ineffective and irrelevant – relegated to being a cost center and unable to help companies innovate or differentiate in highly competitive markets.

Current IT Approaches to Responding to Business

This failure is not entirely IT's fault. In fact IT departments are probably doing the best they can. The problem is simply that current techniques and tools used for delivering and maintaining enterprise systems are hard to use, and they deliver applications that are hard and expensive to maintain.

When the business needs a new application, or new application functionality, IT typically responds in one of four ways. At best these approaches kick the can down the road. At worst, they add to the problem by creating an architecture and technical environment that will be unresponsive and unable to adapt to changing business needs:

Approach 1 Buy, and Customize, a Packaged Application

Consequence: When an organization chooses to buy a packaged application such as SAP or other ERP, payroll or CRM solutions, they are buying into pre-defined functionality that delivers best of breed **standard processes** for the industry/market it serves. More often than not, organizations find that their processes are somewhat different from what the package offers, and therefore must customize the package. The problem is that packages have not been built for change and therefore customizing packaged applications is hard, expensive, can require hundreds or thousands of consulting hours, and will create a maintenance nightmare that possibly hinders future package upgrades.

Approach 2 Build a Custom Solution Using In-house Developers or System Integrators

Consequence: The problem with this approach is that building a large system using the current state of techniques and tools is complex and expensive. In many situations, the organizations then fall hostage to the system integrators who built the application, because they are the only ones that know how to maintain it. In any case, enterprise applications are never "built for change." The urgency and focus on getting an application deployed, using current techniques, produces a solution that is not easy to change, and is often prone to break when you do. This just ends up being a big add-on to existing technical debt.

Approach 3 Select and 'Rent' a Cloud-based Solution (SaaS)

Consequence: With the advent of software-as-a-service (SaaS), companies are resorting more and more to 'renting' cloud-based applications to satisfy business requirements. By the very nature of the model, SaaS applications offer small degrees of parameterization, and with the exception of a couple of very large vendors, no customization options. To a degree, this is a blessing in that IT tends to use SaaS "as-is" and therefore limits maintenance requirements. It does not, however, address the requirement for custom functionality - namely interfaces, workflows and data repositories that are unique to nearly every organization.

Approach 4 Do Nothing!

Consequence: When budget, time or resources are not available to buy or build the needed application, IT often just says "No" to the business. When the IT department chooses to do nothing[4], it causes its relevance to decrease even further **and results in the creation of…**

Shadow IT

When the pressure to come up with a solution for the business is intense, and IT can't deliver what is needed, business teams often decide to take matters into their own hands. This is commonly known as "shadow IT" – applications built, installed or rented outside of IT's control. Typical approaches include using departmental 'wiz kids' or finding hired guns to solve the problem. They may use platforms such as force.com or SharePoint to develop and deliver an application to solve the business problem. Or they may use portal, workflow or collaboration tools – or even deliver completely bespoke applications, built using any number of tools or languages.

With cloud-based offerings delivered as a 'service', the business does not necessarily need to involve IT at all – there is nothing to install, nothing to back-up, no disaster recovery considerations, data conversion can be outsourced – business can solve the problem without IT even knowing the application exists.

However, as these applications are adopted and begin to grow the complexity of changing them, and/or the inherent value of the data being maintained by them, forces the business to turn control of the applications over to IT.

These are the scenarios facing most IT departments when dealing with backlog and KTLO issues, and none of them is particularly desirable. The approaches are replete with cost issues, control issues, and scalability issues. Yet, despite the futility of these approaches, time and again, organizations are choosing to walk down the same trodden paths, expecting to arrive somewhere different.

[4] http://cxo-advisor.co.za/content/abominable-no-men

Redefining the Cause of the Problem: The Cost of Change!

With the problem illustrated, the question IT departments should now be asking – **the question you should be asking is, why?** Why are you continuing down the same paths? It has not worked. It does not work. It will not work. So, why continue? You can race down those paths as quickly as you think you should, but it will not fix your "IT mess." Inefficiency is only a symptom; the cause is something else.

The IT mess you are suffering from is caused, entirely, by the cost of changing software.

Think about it.

Delivering new applications or customizing packages takes a lot of time and money, largely because project teams (internal and outsourced) add a lot of cushion to project timelines and budgets hoping to protect themselves from changing requirements or unpredicted events. They do this because they know that changing requirements in the middle of a project will be hard and will cost more time and money. They do this because they know that the further along the project is, the harder it will be to implement changes. The more code you build, the more dependencies it will have, and the harder it will be to change.

And it never stops, because the cost of change is not limited to the delivery of the first version. As soon as the application/system goes live it will immediately go into maintenance mode, and sooner or later will have to be touched, corrected, extended, tweaked…. Changed!

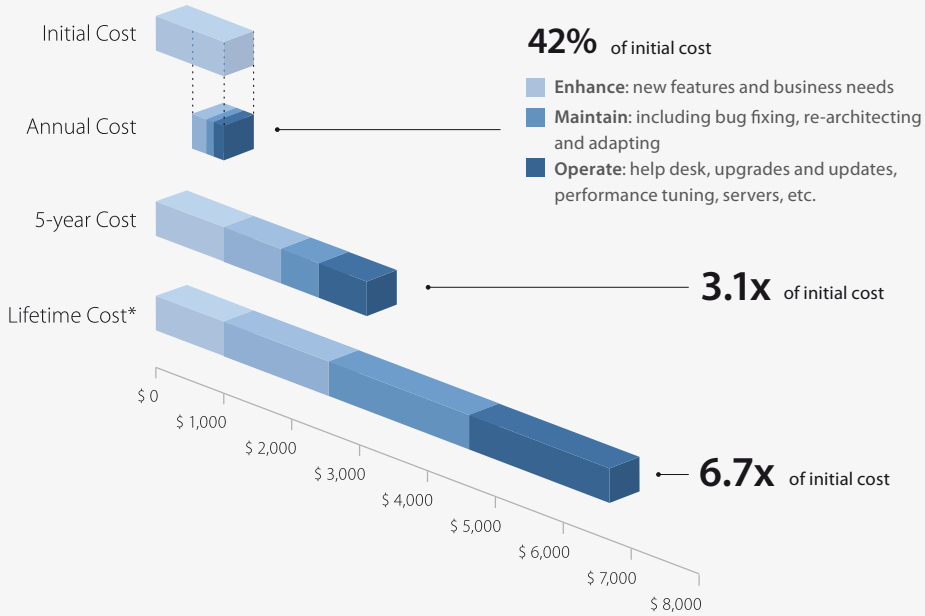
According to Gartner[5], when looking at the total lifetime cost of building or buying a new application, on average 42% of the initial cost of the application is going to be spent, year after year, to maintain that app. **Application maintenance is the real problem.**

[5] http://www.gartner.com/id=1331651

Drilling into that 42% we find that it breaks down into three major buckets. The first bucket is enhancing the application. The second bucket is maintaining the application, break-and-fix, etc. And the final bucket is all the operational costs – the people who run the help desk, delivering upgrades to operating systems and storage environments, etc. These costs are real.

Total Cost of Ownership

Drivers: Amount of Change & Application Flexibility

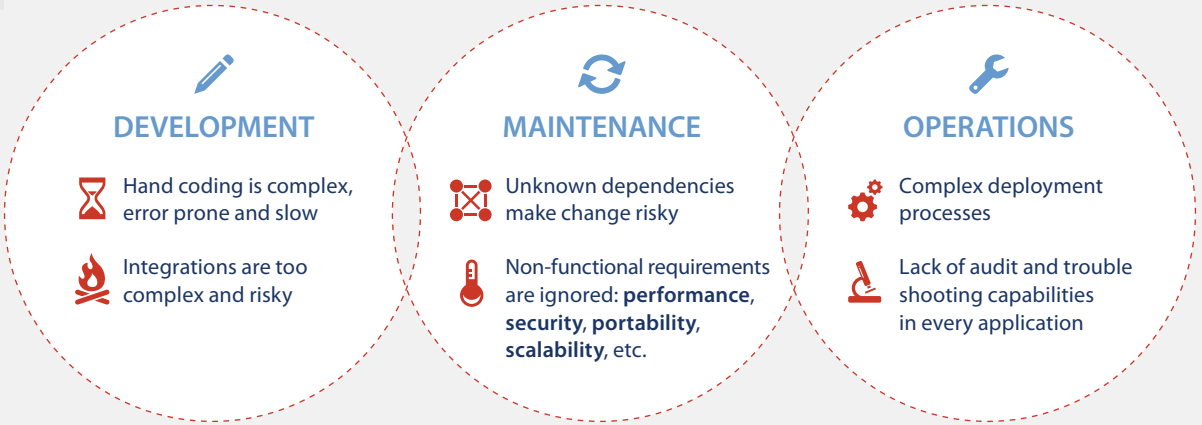


* over a 15-year period. Enhancement costs halved during years 8 to 15

Source: Gartner, "A Framework for the Lifetime Total Cost of Ownership of an Application"

Application Cost Drivers

Application change is the major issue.
As applications grow and evolve, change becomes slower, riskier or even impossible.



Have you budgeted for that year after year cost?

If maintenance costs run 42% of the initial investment, then a million-dollar application built last year is going to cost $420,000 to maintain this year. In the next five years, the business will spend $3.1 million to maintain the application. And if an application is in production for 15 years, it will cost almost $7 million to keep that application up and running. That is one expensive application! Think about how many applications companies have in their portfolio and how many requests they have in their backlog. Every application in your portfolio is contributing to this problem.

The more complex (and older) a system is, the more complex and costly changing it will be. Look at these IT cost inducers regarding change and development:

- Reverse engineering thousands of lines of code to find the exact places where you will need to apply changes requires a lot of time from very experienced (and expensive) developers.
- Validating the impact of those changes across all application and system dependencies is extremely complex because dependencies grow exponentially as the system grows.
- When changes are ready and tested, the process of deploying a new version into production is complex and error prone. You'll need time and experts to make sure it goes well.
- Integrating existing systems is often complex and risky.
- Non-functional requirements are often ignored or forgotten when deadlines come crashing down.
- After going live, no audits or troubleshooting capabilities are in place.

The problem redefined, is that the root cause of your IT troubles has everything to do with changing software, both old and new.

In order to solve this problem, IT departments need to take control of the cost of change. This requires a complete IT transformation where you **redefine your approach to every application in your portfolio and strategically think about how you will respond to business requests.**

Such a transformation doesn't happen overnight. The good news is that it can be done **one project at a time**, with cumulative benefits/savings that can fuel the next transformation phase.

This is a transformation that can take an IT department from being a business scapegoat to a center of excellence and innovation.

A Road Map to High Performing IT

A **High Performance IT** has a clear strategy for each type of application it is asked to deliver and maintain. It understands that the approach and technology it needs to use is different, depending on the functional and non-functional requirements that need to be supported.

As a starting point, we propose an approach to working with applications based on *Gartner's Pace Layered Application Strategy* (see next page).

It begins with looking at your application portfolio and identifying in which layer each application fits best. The next step is selecting a technology to use that effectively delivers and maintains applications in each of the layers, ensuring an adequate pace of change.

With a clear diagnosis, the final stage in your journey is to put an action plan in place. This takes us back to the approaches we described earlier, but now, let us see how they might be thought of differently when we keep the cost of change front and center in our minds.

1 Buy (or rent) software… but don't customize it

If the requirements from the business are "standard processes" that can be found in existing packages or SaaS offerings, go ahead and buy/rent the application, parameterize it, and deploy it "as is". Then make sure that the business adapts to the software and refrains from customization.

Specific functionality required by the business not available in the package (or in the SaaS offering) can then be delivered by creating differentiating web and mobile applications "on top" of your packaged application. Just make sure the software you buy or rent has extensive APIs and that the technology you use to delivery this new functionality is "built for change".

This approach can also be applied to existing packages and legacy systems, which are usually big spenders of maintenance dollars. In these cases, you should stabilize your packages by stopping any attempt to customize or change them. Instead, all change requests should be diverted to new applications that integrate with these systems to access their data and that deliver the needed functionality via new user interfaces, using technology that supports change.

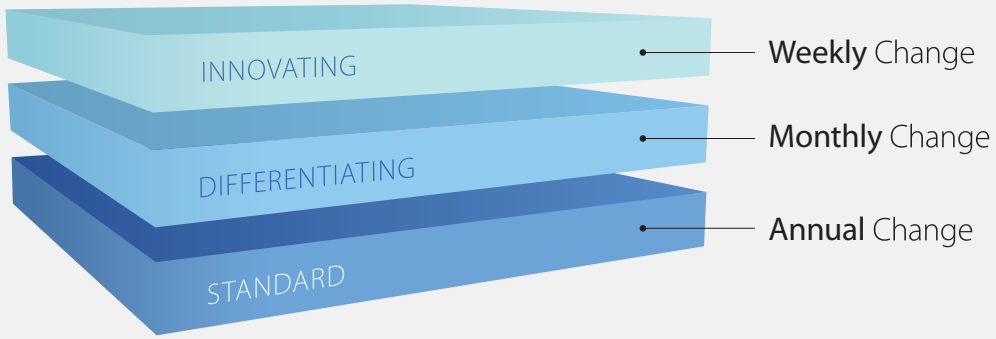
2 Build Custom Solutions… but smarter

Use technology that makes it fast and easy to create custom applications that go from simple departmental apps to complex core systems. The technology you choose needs to allow you to change these apps frequently because they support important business processes that are constantly evolving and will therefore need regular updates. It also needs to ensure that your applications cover the most important non-functional requirements, especially when it comes to security, robustness and scalability.

3 Build Innovative Apps… that can grow

Use technology that allows you to build business applications extremely fast and respond to your business in a matter of days. More importantly, make sure that if the applications become business critical, the technology you select will allow them to grow and evolve without forcing you to rewrite them completely. Remember that many of the innovative apps that are born on the fringes of the company might eventually grow to become differentiating applications that will support your company's unique processes.

Overview of Gartner's Pace Layer Application Strategy



1. The Standard Layer

This layer includes applications that support business processes that are common to other organizations. These are systems that don't need to change too often - annually mostly. In the standard layer, we consider processes such as payroll, financial systems, and even many base ERP packages because these are used in a standard way. They support processes every business has to do; they are mission-critical. If they did not do these processes, then the business would not be able to run. However, they do not add unique value as most competitors conduct these business processes the same way. Applications in this layer are usually big, complex, and were built to last, which makes them extremely hard and expensive to change.

2. The Differentiating Layer

This layer includes applications that support a business' differentiating processes. Differentiating applications typically incur quarterly and sometimes monthly change. As businesses evolve and tweak their processes, the applications that support them must be updated. As applications in this layer tend to be relatively large and critical to an organization's business processes, these systems are the ones that tend to clog most application backlogs because, unfortunately, they are also slow and expensive to change. In the differentiating layer we find applications geared toward core business processes that are critical to business success and provide a differentiated and unique capability from competitors. We find many processes related to order management, customer support or human talent management fit in this layer.

3. The Innovating Layer

This layer is where business innovates. It is where new ideas have the opportunity to be systemized, tested and proven, eventually becoming a business differentiator. Applications in this layer generally require weekly, or even daily change. This is the layer where businesses may provide mobile-type apps, social apps, and any attempt that promotes a new market approach, new channel, new product, or market entry capability. This is where businesses work to capture a unique piece of the market with an innovative idea. They usually begin with technology like email, Excel spreadsheets, an Access database or even SharePoint. However, at some point in time, if the business process proves to be working, those applications are going to become an IT problem because they are going to need to scale. They are going to need security. They are going to need all the functional and non-functional requirements that a true enterprise app has that people do not typically worry about when they are just trying to invent and test a new process.