APPLICATION & PLATFORM LEGACY MODERNIZATION: BENEFITS & RISKS

What is Legacy Modernization?

Legacy modernization is essentially to transport existing or traditional software functionalities to a context that is compliant with up to date IT landscape potential. The most obvious example is moving process from a traditional on-premise data center model powered by mainframes to a public cloud environment rich with microservices and containers.

When modern technology typically feels obsolete soon after its debut, the mainframe is an anomaly as there are still thousands of companies at the global scale that have critical core business processes based on corporate software that dates back some 30 or even 60 years. Even with its long history, mainframe transaction volume increased in 59% of mainframe shops according to the 2018 BMC Mainframe Survey of 1,100 IT executives and professionals.

COBOL, the widely used language for mainframe programming, debuted 60 years ago, and is still being used by clients on IBM's latest mainframe hardware.

Given the age of the mainframe, conventional wisdom says it's ripe for replacement with more modern systems and platforms. While there are many benefits to moving away from legacy or traditional systems, there are also many cons that must be weighed before such a move is made.

Benefits of Legacy Modernization

Having such old software tools supporting business operations potentially means incurring into multiple growing risks, costs as well as critical constraining factors. Here are some examples:

Obsolescence

- **Losing control**. Having a part of one's core business dependent on some piece of code developed in some "old school" programming language whose brain trust is now in or approaching retirement has the power to render sleepless the more relaxed business managers.
- **No support**. Having any type of critical business processes running on hardware and/or operating systems that are no longer supported by their manufacturer or there is even no longer a manufacturer to resort to; well, hardly the position any business manager is eager to be in.
- **The missing component**. The support infrastructure no longer exists. Some older software was developed having its backup process and policies dependent on either the backup hardware or the logic inherent to some specific backup product or solution. Most likely such backup product has already been discontinued which potentially renders restore activities unfeasible.

Cost

- **Choosing the lesser problem**. If a Legacy system is still around it usually means that it plays a critical role towards corporate core business. In most cases, replacing it comprehends significant costs both investment-wise as well as regarding potential operational stoppages and other related running costs. Still, doing nothing, although it may seem to save money in the short term, may represent an exponentially growing risk of not being able to work at all in the foreseeable time horizon, particularly if the hardware or operating system are no longer supported by a manufacturer.
- **OPEX and CAPEX**. Having old systems running may imply investing in discontinued components stock such as backup tapes, hard disk drives, RAM memories or resorting to high-cost contractors that leverage their niche knowledge.

Agility

- **Responsiveness**. As more burden is placed on digital systems to support customers and business critical processes, some legacy systems may be causing a bottleneck and slowing over application responsiveness. It's important to examine if it's an input/output issue or if it's a processing issue that's causing the issue.
- **Business Cycle**. Being competitive in today's market requires the ability to promptly address an unforeseen demand or release new features quickly. Some legacy and traditional systems may not able to cope with the required flexibility and speed to deploy new code quickly and efficiently.

Integration

- **The ability to easily speak to others**. IT has been growing in terms of platforms served (mobile and social networks) and complexity in terms of IT landscape components (cloud, hybrid and on-premise data center). Agile "plugin" integrability is becoming a competitive edge in business terms and Legacy Systems were developed with a more siloed mindset.

Misalignment

- **The Law**. In some cases, legislation or market regulation may imply the need of migrating an existing software so that it may comply with newly posed requirements.
- **The Market**. Competition is another common compelling motivation towards Legacy Modernization. Upstart competitors are typically not weighted down by technical debt, older systems, and processes, making them nimbler by nature. When competition gets ahead in the market by having more effective business tools, the time has come to move forward quickly or perish.

Beware: Newer is not always better

Legacy modernization may seem like an easy and logical decision when dealing with systems that are multiple decades old. But as the old adage goes, "If it ain't broke, don't fix it." A new paper from Gartner Research warns that moving away from legacy systems such as the IBM mainframe could end up costing more and pose a risk to quality.

The paper advises organizations considering a legacy modernization project to first:

- **Focus on business needs and capabilities, not perception.** Phrases like "fragile" or "legacy" – or even Gartner's preferred term "traditional" – can carry connotations and the language can bias decision makers towards modernization when it isn't necessary or even beneficial.
- **Audit existing platforms and processes** with a focus on misalignments or gaps between business requirements and what the platforms deliver. Start with what you've got, not with the assumption that change is necessary.
- **Consider total cost of ownership,** including cost of transitioning and dependencies, over multiple year period. By focusing on business needs and cost of ownership, organizations can correctly prioritize projects that will drive the most business value while minimizing risk.

Considerations for the cost of such a move need to go beyond current CAPEX for the traditional system versus expected OPEX of a new system. Businesses need to also look at things like change order costs, the cost of running multiple systems simultaneously as the transition is being made, training costs, and any new security exposure.

The emphasis is to look at the issue from a business perspective and ask questions that relate to the problems with competitive ability, current backlog of change requests, friction in the business workflow, and where failure occurs. As Gartner simply puts it: "The objective is to determine whether the traditional platform is helping the business or hindering it from meeting its goals."

Popular approaches to Legacy Modernization

The entire topic poses such a complex multitude of potential impacts that several approaches and methodologies have been developed just to address it:

- **Architecture Driven Modernization (ADM)**. Resorting to support infrastructure to mitigate the modernization complexity (virtualization is an example).
- **The SABA Framework**. Consists of planning ahead both the organizational as well as technical impacts; in fact, it is a best practice that if largely used would have minimized some headaches in big corporations.
- **Reverse Engineering Model**. Represents high costs and a very long project that may be undermined by the pace of technology.
- **Visaggio's Decision Model (VDM)**. Consists of a decision model that aims at reaching the suitable software renewal processes at a component-level based for each case combining both the technological as well as the economic perspectives.
- **Economic Model to Software Rewriting and Replacement Times (SRRT)**. Similar to the above mentioned VDM model.
- **DevOps contribution**. DevOps focus is to allow swift deployment of new software releases with an absolute minimum degree of bug or errors in total compliance with target operational IT environment. This, by itself, represents a major enabler factor to speed up Legacy Modernization processes.

When considering Legacy Modernization there are three possible roads to choose from:

- **Rewrite**. Migrate the coding to a more recent development language. This option implies a high potential for underestimating both inherent costs and required time frame, therefore running into a never-ending story.
- **Replace**. Throw away the old system, replacing it with a new one. This option usually implies having to build an entirely new application ecosystems with semi off-the-shelf components that mimic or mirror some functionalities of the existing Legacy Systems, therefore usually meaning raising the complexity level without, most of the times, achieving 100% functionality compliance.
- **Reuse**. Find a manner to achieve portability of the Legacy System as is to an up to date IT environment. If applicable, this option allows a step-by-step approach where change is tested and fine-tuned.

Independently and prior to deciding which is the most suitable way to proceed, it is vital to have a clear picture about current status as well as of how well are the foundations established.

Getting started with Legacy Modernization

Here are some steps that must be accomplished in order to layout a Legacy Modernization Process roadmap that results in added value:

- Clearly identify and map all IT Systems and inherent software applications within one's corporate landscape, regarding role, interactions (both amongst them as towards human users), their support platforms, resources and ageing status within the current technological standard.
- Clearly identify, list and mirror towards existing IT Systems and inherent software applications

the corresponding corporate business processes.
- You must look at the business as well as IT and if the company does not have such culture and still considers IT "something annoying that we must have," you must need the unequivocal sponsorship of the board of management.
- Proceed with a dual approach perspective considering in one hand the IT Systems and inherent software applications technological ageing status versus current technological standard, and in the other hand matching them against corporate business processes evolutive roadmap.
- The overlapping of both timeline maps should allow a good perspective on required Legacy Modernization prioritization roadmap.
- Bear in mind that people cannot provide you what they do not have, meaning some process knowledge needs to be checked and cross-checked because most of the time although key users know how the process works and which steps it encompasses they do not know the logic behind it nor the implications in other business processes.
- Carefully cross check the Legacy Modernization prioritization roadmap with IT systems interdependencies and fine tune the roadmap accordingly.
- Raise the core business potential impact while performing a risk assessment regarding each Legacy Modernization action and redefine the roadmap accordingly.
- Raise the budget impact of each Legacy Modernization action and redefine the roadmap accordingly.
- Prepare a final roadmap proposal and have it clearly conveyed (along with inherent risks and benefits) to all corporate stakeholders getting their buy-in.
- Have a clear failover strategy and make sure to also convey it in an assertive and crystal-clear manner.

IT systems are one of the main pillars of modern corporations and therefore Legacy Modernization deeply affects the company at all levels. It is something that just cannot be looked upon as the CIO's problem. It's either a benefit or potential catastrophe that mandates a corporate approach.

In some cases, the potential impact of attempting software migration bears such a risk that CIOs and top executives are left with the sole option of delaying those projects, hoping for technological evolution that enables lower cost and lower risk solutions.

Waiting for the next big thing might be wiser

Some seven years ago the evolution of virtual environments allowed some old, yet critical IT systems to be maintained as they were developed, just having them migrated to a virtual environment that basically holds them within a more secure and redundant "protective capsule." But, unfortunately, not all old software are good applicants for such processes.

Risks, challenges, and benefits of Legacy Modernization

Inherent risks to migrating Legacy Software include:

- **The challenge of human change management**. Humans deeply dislike change and need to be motivated, trained and coached towards it, which represents additional risk and cost.
- **Old and new cohabitation**. Most corporations are not limited to one single Legacy System and having the several existing ones migrated simultaneously is just not feasible due to the

potentially catastrophic impact combined with the imminent coordination effort and costs. So, in most cases Legacy Modernization needs to be a corporate program that looks at each system required effort and time window as well as how to best articulate and prioritize the migration endeavor.

- **Tailor-made philosophy**. Legacy applications were born in an era where the main concern of software engineers was to develop code in a way that better explored the specific platform (hardware and operating systems) where the application software would be running. This means that when migrating a Legacy System or application one extra care to be considered is how to tackle the assumptions that laid beneath the coding decisions taken to better explore specific platform functionalities that no longer exist, either having been discharged or become obsolete by new technology. One clear example relates to former 68000 CPU based hardware that would leverage RAM indexing in a manner that simply isn't relevant any longer; nevertheless, such strategic approach to coding may render ineffective a direct migration effort. Another example relates to the way in which former Novell Networks would allow information packages to flow. When Novell was heavily replaced on the 1990s, some applications would not work because they had been developed to have their I/O optimized according to such Network protocols.
- **Integrability cycles**. In a corporate environment there are more systems "speaking" to each other than with humans. This means that one main concern when modernizing a piece of Legacy software is to assure that I/O will obey by the data interchange rules and requirements posed by client applications and support resources (like databases).
- **Hardcoded business processes**. Let's consider tens of thousands or hundreds of thousands or even millions of lines of code that have been developed within a given logical architecture because it better addresses a given corporate process. Those lines of code have most likely been recurrently "enhanced" over several years via the intervention of the team that knew the underlying logic behind the entire software coding. To end this story let's consider that such team is gone and absolutely no documentation is available allowing to understand how the software matches the corporate process. It is a potential trial and error nightmare, aggravated by the fact that in most cases such Legacy Systems may not be replicated in "laboratory" therefore change needs to happen live, with the potential core business impact that derives from that.