GROWTH ACCELERATION PARTNERS

MODERNIZE, REWRITE OR REPLACE? HERE'S WHAT THE DATA SHOWS TO GET IT DONE RIGHT

LEGACY APPLICATION DEVELOPMENT

Nothing lasts forever. And often, that includes the skeleton in your company's closet: a legacy applications in desperate need of getting brought up to date. Status quo is a tough barrier to break through. But whether it's a philosophy of "if it ain't broke, don't fix it" or an ingrained cultural inertia holding you back, rest assured you are not alone.

That legacy software application your IT team wrote in the 90s has been chugging along for almost 30 years. Apps written in a decade when the Internet was nascent — when platforms were limited and users would take what they were given — are obsolete (or worse) in the 2020s.

What to do? Modernization is often cited as the preferred mitigation strategy, with market research firm Statista estimating a total market value for modernization tools at $8B in 2018 and predicting it would hit almost $37B by 2027.

That's a lot of CapEx, and for a very good reason: legacy applications are a significant and broadly recognized problem across the organization, whether you measure security, competitiveness, agility, efficiency or staffing. And the risks associated with not keeping up-to-date are evident and numerous. Once a system gets to end of life, your application can be severely compromised both in performance and security. What's more, the longer you wait for an upgrade, the more difficult it becomes to execute.

In some cases, legacy iterations are so incompatible, no clear upgrade path exists. And that simply won't work when you need something mobile-friendly and compliant. Maybe the IRS can survive with a system written during the Apollo program at NASA... but can you?

DOOR NUMBER 1, 2 OR 3?

Choosing the modernization path is not only the most popular, but the most effective. Why? Because many legacy projects do not necessarily involve old technologies; just older iterations.

One analysis firm that has tracked a LOT of software development projects, has polled over 50,000 participants over multiple decades and found "modernization" beats the other two choices — rewriting from scratch or replacing with commercial off the shelf software (COTS) — by a wide margin. In fact, 71% of respondents reported successful modernization projects, while only 26% of the "rewrites" were successful, and a similar result (27%) for successful COTS "rip and replace" projects.

Some software was written in a language no longer in vogue or even supported (think Visual Basic classic, PowerBuilder and COBOL), while other programming languages have evolved and left the older versions incompatible with current releases.

PROBLEMS WITH PYTHON

Take Python, for example. The last iteration of Python 2, Python 2.7 — released in 2010 — was out of support as of 2020, but it's still around. Yet even though the latest version is as far advanced as 3.11, organizations continue to cling to the older iteration. And Python 3 breaks a lot of Python 2 code, so upgrading is more than just a re-compile.

While Python 3 has easier syntax compared with Python 2, this came at a cost of removing backwards compatibility within many libraries. A very simple example of this subtle change is in the print function, which presents the output of a specified message to a screen or device. Print in Python 2 is a statement, but in Python 3, it's a function requiring parameters. The parentheses are necessary to wrap the object you want to print — and using it as a statement will result in an error. It's not a huge change, but spread across a zillion instances in a big app, this is a lot of work to fix.

Syntax highlighting can reduce a developer's workload by about 20% in this regard, yet a major release of a programming language represents a significant change, from declaring the code, to handling imports in different libraries among others — and developers need to be able to adapt to this.

Even modernization/upgrade projects that look simple on the surface can hide nasty bits.

During a project for a U.S. technology company that primarily sells computers and related products/services, GAP upgraded the client from Python 2.6 to 2.7, in what seemed to be a non-breaking upgrade. But unbreakable does not mean the same as seamless. The project required sourcing a new database connector to work correctly, and that in turn required different testing tools and redundant test coverage.

In another GAP project, a client with an application management system for scholarships, grants and awards had a longstanding application written in Ruby, with Ruby on Rails as the framework. Unsurprisingly, an upgrade was recommended. Upgrading the language crashed on the older framework, and updating the framework broke all the dependencies. But with the finished solution, performance improved significantly: typical load time for the application changed from approximately three seconds to half a second.

Experience is the key to predicting and planning correctly for all the subtle issues that will arise in a modernization project. The first time you drive a road, the potholes are a surprise. After enough trips, you know how to avoid them.

VISUAL BASIC 6 HAS ENTERED THE CHAT

When GAP acquired the application modernization business of Mobilize.Net in February 2023, they scored the most used and most complete set of tools for automated conversion of ’90s era source code, like VB, PowerBuilder, ASP.NET and more.

Today, under the umbrella of GAP Mobilize, customers can get the best-in-class efficiency of AI-assisted automation in their modernization efforts. Whether you need to move ’90s desktop or early Web architecture to modern languages (like converting VB6 to C# or VB.NET) — or even migrating desktop apps all the way to native web apps using ASP.NET Core and Angular — GAP Mobilize has tools, services and expertise to reduce time to market, overall risk and total cost.

There is no magic trick to application modernization – but the tools from GAP Mobilize come close.

RECREATIONS AND AUTOMATION

When it comes to migration of data, there are some other pitfalls that may come into play. A theoretical example of this would be for a project in .NET. It is common to use stored procedures, which are procedures written in a .NET language containing SQL statements, and they can be considered similar to a function in programming. If you were to keep using these stored procedures while upgrading your database from SQL Server 2005 to 2019 — a major upgrade — then chances are the majority of your search materials will not work. And while the data will remain saved, these store procedures would have to be recreated.

For recreating the application from scratch, with an entirely different structure, a chunk of the job would be to convert data from the old version of the application to the new version. With a large data set, that by itself is a huge project requiring a lot of automation.

No matter who you are, setbacks can occur at any time. Organizations need to not only have a strong plan and timeline in place, but leeway in place to afford and allow for a longer timeline. A defined migration timeline could be six months; the migration might take nine months, and the whole project might take a year to complete.

SUCCESSFUL SOLUTIONS

One other factor to consider — and a significant reason why you may want to rely on third-party expertise — is with regard to the sheer resources you need for a migration project. If you are putting together the new version of the application, you still need to update the old one if your customers are still using it. Maintaining two versions, including frameworks and code, will stretch a small development team to breaking point.

You could use a pool of engineering talent to create updates for whatever language your application is written in, and also provide expertise in cross-disciplines such as QA. Good test coverage and planning, as well as strong QA scenarios, are key for a successful migration.

Application modernization and migration is hard, and there is no one-size-fits-all solution. But GAP has a wide range of expertise, both in breadth of languages ranging from Java to .NET, and Ruby to Python, and also in delivering different types of client projects. Whatever the technology, whatever the brief, we have the experience to cover it.

Furthermore, GAP has earned the trust and respect of clients over the years because they are known for creating and executing the best path forward. And you can be sure GAP will navigate you to the other side, safely and quickly.

09262023

To find out more, please visit WeAreGAP.com

company/growth-acceleration-partners/ @GrowthAccelerationPartners @GAPapps