2018

ROI of PowerBuilder Migrations

REASONS AND BENEFITS OF MODERNIZING

MOBILIZE.NET

Summary

Thousands of PowerBuilder applications still run today. Even though the platform is supposedly supported, its development has been slow in the past decade and the platform has changed hands multiple times which has caused the technology to become less desirable and lack basic functionality from more modern platforms.

This paper address issues with continuing to use and support PowerBuilder applications and examines alternatives for potentially disposing of them. Among the many reasons PowerBuilder is no longer a good thing to rely on are:

• Security issues

• 32-bit performance issues

• Potential to lose customers because you're a dinosaur

• Developers and resources are expensive and hard to find

• Can't do anything modern like take advantage of SaaS interfaces

(i.e. can't integrate with Salesforce or other SaaS products, etc.)

• It's an obsolete technology with no real path forward.

Disadvantages of PowerBuilder

This section mentions several drawbacks of maintaining PowerBuilder applications.

Orphan technology

How much can you trust the maintenance of a platform that has changed hands three times in the last decade?

The truth is that PowerBuilder doesn't seem to have a clear path forward and it has been changing hands because of the IP more than to continue developing it as a development platform.

This causes the community around the platform to reduce and the availability of resources for it to diminish as well. As an example, if you look at questions tagged "PowerBuilder" in the popular StackOverflow.com forum (one of the most popular portals for programming, in every technology) the number is just over 1,000 questions, while searching for technologies such as Spring or ASP.Net brings up 140,000 and 340,000, respectively.

Even in the Tiobe Index (https://www.tiobe.com/tiobe-index/), a recognized index of the most popular programming languages in the industry, PowerBuilder stopped being listed in the top 100 languages in the year 2009, while languages such as Java and C# are both in the top 5 places.

Lack of Developers

A common comment from companies that reach-out to us with PowerBuilder applications is the lack of resources to maintain (let alone add features) to the existing applications.

PowerBuilder was a popular platform in the 90's but its popularity went down from there and nowadays it's hard to find experienced developers for this technology. In many cases the original developers of the applications retired or left the company and now the applications (even though they work) are static and the existing development team of the company dread having to go to this platform to perform any changes.

Moving to a modern platform gets rid of this problem as the technologies targeted by Mobilize's tools are modern and very popular which means access to new developers is easier.

Tied to a Desktop Environment

PowerBuilder was a great platform to build desktop applications but for many scenarios the users expect applications that can be used from multiple devices and platforms.

Migrating to standard Web technologies will allow users to interact with the applications from any device that can render HTML (e.g. mobile phones, tablets, etc.) and will also allow administrators to simplify the deployment process of these apps.

Hard to take advantage of new technologies

Technology is evolving constantly and some of the best and most exciting improvements over the last few years can help any business save on costs and help with the quality and performance of their applications.

Many of these improvements are easily accessible from new platforms but very hard to consume from a platform such as PowerBuilder. Things as basic as consuming a RESTful API require the use of large

amounts of code and little documented libraries. This functionality is natively available in modern technologies and can really speed up the development process.

Similar things happen when trying to use tools for automated testing, continuous integration or profiling of applications. Many of these can significantly improve the stability and productivity of development teams and just cannot be used in PowerBuilder.

Modernization Benefits

In the previous section it was made clear that there are many reasons why PowerBuilder is problematic, however, is it worth the effort and cost of upgrading to a new platform? After all, it is an application that has been working just fine for the last few years, right?

The answer to that question is probably "depends", and it certainly will be based on what the applications do and what the plans moving forward are. Alternatives to help you get out of PowerBuilder exist (more on that in *Getting off PowerBuilder* below) and here are some of the best reasons to do so.

Access to new technologies

Technology evolves daily and better technologies are emerging all the time. Things that are common in new applications weren't even prototypes 20 years ago when PowerBuilder was popular.

These new technologies can help many applications and only by moving to a more modern technology can they be fully utilized.

- **Cloud computing**: This is a broad term that encases many technologies that are offered to decentralize a company's infrastructure and have access to managed resources that in the past would have been very costly for a company that didn't specialize on this.
  Things such as serverless functions, RESTful APIs, elastic computing and access to practically unlimited scalability can be used easily after the PowerBuilder applications are modernized and can reduce deployment and maintenance costs and allow new business models such as Software as a Service and effortless scalable applications.
- **Data analytics**: Moving to a platform that allows an easy instrumentation of the applications allows developers and business stakeholders to have new insights of the ways in which their applications are used. This can be used to help drive the development of new features and be proactive with the changes to the applications.
- **New development tools**: The tools available to use all these new technologies have improved tremendously in the last decade and this reflects in time saved by the development team when implementing new features and doing all their day-to-day work.
- **Automation**: Along with new development tools other improvements have been made to increase productivity during the development process. These include new (and better) tools to automate test cases (QA for applications is estimated between 30% and 50% of any software project so any time savings there will have a huge impact on costs) or to perform repetitive tasks such as continuous integration (automatically build, test and deploy applications after every code change).
  All these can be huge time savers which reflect on overall savings for the company.

Reduced deployment costs

A common nuisance of desktop applications is their deployment. Creation of installers can be a tedious process and verifying that it works on multiple environments adds to the cost of the project.

Once the application is ready for deployment then there's the issue of having all the users install the latest version. This is not easy to do and in many cases the development team needs to give support to multiple versions of the application. All of this goes away with moving to a Web platform.

- **Single server deployment**: Applications can be deployed in a single server or farm of servers and accessed by all the users at once. Everyone will always be using the latest version and by taking advantage of the cloud this can grow or shrink automatically based on real-time requests from the users.
- **No remoting licenses**: Some companies use technologies such as Citrix or RDS to help with the deployment issues by having all the users connect to a remote desktop in which they can control the desktop application.
  This can work but has several drawbacks:
  - The licenses for these components are expensive.
  - Users require many more resources than when using a Web application
  - Giving remote access gets the people in the local network which can pose security problems
  - The scalability of this solution is not as simple or documented as with web servers

Access to more developers

As discussed in the Disadvantages of PowerBuilder, the access to qualified resources in the new Web platforms is easier than hiring PowerBuilder developers. The new technologies are also easier to learn, and developers will have access to better resources if they need to be trained on them.

Using new technologies opens the possibility of increasing the team faster but also with cheaper resources (interns or entry-level developers) which will be hard to find for technologies that usually have been used only by developers with 15 or more years of experience.

Finally, many developers are also interested in going to or staying in places where they get to learn new technologies and can use all the new things that they read about or hear from other colleagues or the conferences they attend.

Being able to hire people who can work on these applications is key to continue evolving them and is very hard to do if the applications stay in PowerBuilder.

New Business models

For many companies the use of PowerBuilder not only limits them from a development standpoint but also from a business standpoint. Things that weren't as popular a decade ago can now make-or-break a company and the use of new technologies can help move in that direction.

SaaS

Many companies have started offering their products with this model where multiple clients can share infrastructure resources, allowing an easier maintenance and lower costs which enabled models of monthly payments and access to a new market.
Staying as a desktop application requires incurring in additional support and deployment costs which makes accessing these markets next to impossible.

Mobile Apps

Another model that is very popular due to the pervasiveness of cell phones is the creation of mobile apps. In many cases these can be created as an add-on to the full-fledged applications to access some functionality such as overviews or basic data query but not for the full reporting functionality or complex forms.

The advantage of moving the existing functionality to a Web platform is that very easily this functionality can be isolated and exposed to be consumed by mobile apps or even the UI can be customized for smaller form-factors and re-packaged as a mobile application. This is clearly impossible with a PowerBuilder application.

Security & Compliance

One of the most important things for many organizations is making sure their data is protected and therefore standards for security and compliance are put in place.

A basic component of these protocols is the application of the latest security updates and making sure the platforms in which the applications are built not only are up to date but also supported so that if a new vulnerability is discovered it can be patched and the problem addressed in as little time as possible.

Using a modern platform with a big user base such as .NET or Java ensures many years of such updates; as opposed to a technology such as PowerBuilder.

Getting off PowerBuilder

You don't have to live with PowerBuilder forever because there are perfectly valid alternatives to remove the risks and costs of having PowerBuilder. Here are three popular approaches to remediation with pros and cons:

Choice 1: Replace with off-the-shelf software

If you are an ISV and you sell your PowerBuilder app, then obviously your app is custom to your needs. But if it's an internal (i.e. line of business) application, perhaps you don't need a custom app to solve the problem. Platforms such as Microsoft Office or Salesforce can be extensively tailored to perform business functions that in the past required writing software.

Pros

- High quality built-in
- No in-house development resources needed
- Rapid deployment
- No large up-front investment required.

Cons

- Lose competitive advantage of custom features
- Perpetual dependency on an external vendor for mission-critical functionality
- Per-seat costs can mount up over time
- Can be difficult or impossible to customize completely
- Requires retraining of staff to new paradigm
- At the mercy of the vendor for support and bug-fixing.

Choice 2: Rewrite from scratch

When developers look at old PB apps, probably the first thought that comes into their minds is "rewrite."

And certainly, this is a perfectly valid outcome for many legacy applications, but not always. As a company that has seen more PowerBuilder legacy applications—large and small, ISV and line of business—we can attest that there are many examples where a rewrite is a terrible idea. Let's drill down.

Consider that many PowerBuilder apps were built around the concept of "forms over data" with some simple business rules enacted in the code. Those apps have little unique value that couldn't easily be duplicated. Their primary function is CRUD plus reporting; that is create, read, update, and delete records in a database and present views of that data. Simple business rules like data validation are built into the code behind form objects.

Small apps that follow this model with dreadful code (buggy, tangled, hard to follow) are best rewritten from scratch or replaced with some package that can do the same work.

What kind of applications should be rewritten? Probably far fewer than most people would expect.

The reality is that writing software—any kind of software—is incredibly risky. Numerous studies have shown failure rates of up to 70 percent for new software development, with typical problems including:

- Time: it takes much longer than planned
- Budget: it costs more than expected
- Scope: requirements and specs keep changing and increasing during the project
- Defects: more found and not found than expected
- User acceptance: upon delivery, users reject the app for issues such as performance or completeness
- Outright failure: after multiple setbacks, the project is abandoned

Pros

- More fun to write new code than work with old code
- Get to start with a clean sheet of paper
- Can use any language or platform desired
- More appealing to developers.

Cons

- Cost: writing code is expensive
- Investment: lose any prior investment
- Risk: see above
- Time: rewriting a large app can take years to complete.

Choice 3: Migrate with automated tools

Migration using automated tools can be the perfect solution for moving many kinds of PowerBuilder apps forward. Using automation reduces the time and cost of creating a modern version of a PowerBuilder app by as much as 90 percent, allowing you to get your new app into users' or customers' hands quickly and affordably.

Best-in-class migration tools use machine-assisted learning and AI algorithms to analyze code patterns, not just syntax, creating new code that recreates the intention of the original application. In doing so, it preserves business logic, rules, and data structures without introducing new errors.

Which apps benefit most from migration?

Not all apps are good candidates for automated migration. Those that benefit the most have the following characteristics:

- Large and complex, with many forms incorporating complex business logic that is critical or represents competitive advantage
- Has many users
- Is frequently updated and deployed to user community.

Migration is not a rewrite

A migration doesn't do what a rewrite accomplishes, but that's not necessarily a bad thing.

A rewrite addresses the same business problem as the legacy app, but invariably gets additional requirements from the business owners. A migration, on the other hand, perfectly preserves the existing functionality while moving the code base forward to a modern language and platform.

A rewrite is "let's start over."

A migration is “let’s lift and shift.”

In many ways migrations are superior to rewrites:

- A migration is that rare case in software development: a perfect specification, since the existing legacy application is the specification for the migrated app.
- A migration can be risk free by contracting with a vendor for a fixed-price, guaranteed delivery date project.
- Quality in a migrated application is simple to prove: it merely needs to pass a test suite based on the original application’s functionality. When the new app functions identically to the legacy app, it’s done.
- A migrated app reduces to zero or near-zero the number of logical defects in the code.
- A migrated app can be built in such a way as to have virtually no user impact, insofar as it will have the same look and feel as the legacy app.
- A migrated app can be deployed in a fraction of the time necessary to re-create the app via rewriting.
- Once migrated, the new code base can be refactored and enhanced to bring new capabilities to the application.

Conclusion

If you've read this far you know there are real consequences to continuing to rely on PowerBuilder applications. And you know the future continuity of anything built with PowerBuilder —and the associated implications of that—is unpredictable and risky to count on.

Why not start today exploring your path to freedom from PowerBuilder?

Mobilize.Net—the world's authority on automated software modernization and how to get off it—is here to help. Call today.