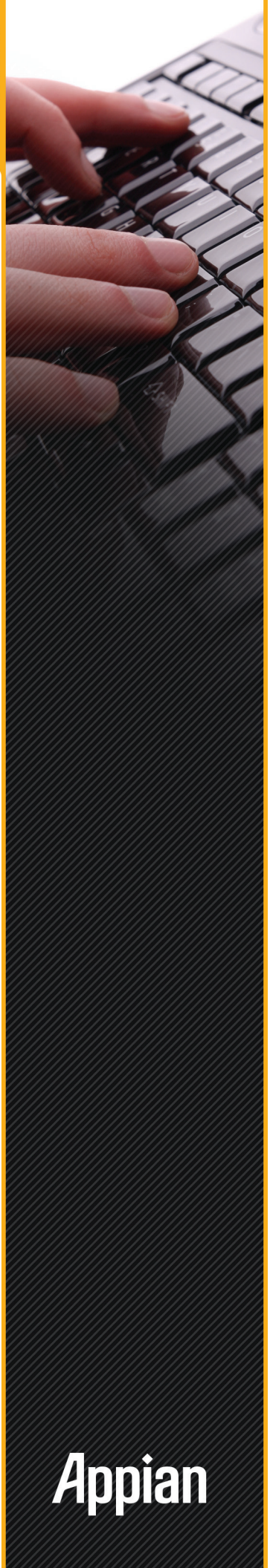
WHITEPAPER



Why Licensing Another Software Application Could Be Costing You Long-Term Success

By Evan McDonnell

Limitations in primary applications like ERP systems create gaps that have to be filled. Licensing specialized software applications is one approach, but it has two big problems. One is that those applications usually target only a single issue so you need to license a lot of different applications. The other is that control of those applications rests with the vendor, not you, so it's unlikely they will completely fit your needs. The ideal solution is custom applications that address each problem and fit your business requirements perfectly. But the cost of development makes this uneconomical for all but the most critical issues.

New technology exists that can end your company's dependence on single-purpose applications and give you the best of custom-developed software. This technology demolishes three assumptions about application development that had been considered bedrock: 1) applications can only be built by developers writing code, 2) full specifications need to be written before development can start, and 3) making changes to applications once complete is cost prohibitive.

Adopting this new breed of technology will help your business users adapt quickly to change and continually refine and improve your operations. It's the key to increasing agility in a world where *"only the agile survive."*

INTRODUCTION

"Why am I getting requests for more new software tools? I thought I made it clear that we already have too many software applications and need to reduce our number of systems. These things grow like weeds."

- CFO of a multi-billion dollar manufacturing company

The team at Appian hears comments like this frequently, so it's likely your CFO or other senior executives have said the same thing to you.

Appian

One of the driving factors behind this frustration is that ERP systems have clear limitations, especially in managing the inevitable exceptions that fall out of regular processes. These limitations often result in people enacting manual work-arounds. When a targeted software solution comes along that can stop the manual work, it seems logical to bring it in-house. The problem is you end up buying **a lot of different applications** as each is designed to only tackle **a single problem.** The growth of cloud solutions compounds this problem, as teams in your organizations can move forward without centralized IT help. Making things worse is the fact that none of the applications work exactly the way you want them to. That means your company must continually adjust its practices to fit your software. ***Shouldn't technology adapt to our needs? Not the other way around?***

In my job I get to speak with senior leaders across a diverse range of companies. The single-purpose applications they are buying are a big improvement over the alternative, which is managing work manually with printouts and tracking spreadsheets. But while single-purpose applications are all good by themselves, the summation of all of them has become a significant problem. Today we're living in an environment of "*application sprawl.*"

The costs of sprawl are significant as each new application has a high marginal cost. Every time an organization shops for a single-purpose application, there's a cost to research options, run an RFP process, allocate hardware, educate a subject matter expert, and train IT and business users. Ongoing updates and upgrades require additional resource investments.

The costs and inefficiencies of single-purpose applications could be tolerable if each was a perfect fit for your needs. You might feel that way at the beginning of a relationship with a new software vendor, especially a young company. But the software vendor's need to serve a growing client base forces them to choose winners and losers among their customers. The "product roadmap" is the vehicle by which they telegraph which customers they will make happy and which will no longer have their needs met. You have two equally bad options if you see your needs not being addressed on the roadmap. You could switch to another software package, but the new license fees, retraining costs, and operational disruption are significant. Or, you could enact work-arounds to deal with your current software's limitations. But that will add to your operating costs and reduce your organizational agility.

The collective frustrations of business managers everywhere facing these problems are sowing the seeds for a new software revolution, one where "application personalization" becomes the new standard and *business users* are able to create *their own software applications to meet their company's needs*. The "product roadmap" will no longer sit with a software vendor. It will rest in the hands of your team responsible for the process the application is designed to help. The roadmap is also likely to become an artifact of the past as software changes are so easy to make in this new world that it will take less time to implement them than it will to write them down in a formal plan.

HOW DID WE GET HERE? (THE LIFE OF AN OLD-MODEL SOFTWARE COMPANY IN SIX ACTS)

I want to help you see the new world this revolution is creating as clearly as I do. Understanding it will change your view of the spending requests sitting on your desk and allow you to make your company substantially more agile and competitive. Those who don't understand history are doomed to repeat it. So let's start out by gaining a clear view of how single-purpose software applications go through their life. It's a story I've seen so many times I think about it in the context of a play on stage.

Act I – A software developer finds a company struggling with a problem. The company asks him or her to create a program to fix it. The company is happy with the fix, and the developer walks away, with an idea…

Act II – The software developer starts asking around and finds lots of other companies with the same problem. This kick-starts the entrepreneurial process. Deals are made, license rights acquired, venture capital funding is raised, and a sales team is hired. More customers sign on and become equally happy…

Act III – Success breeds more success, but with a price. Each successive customer has slightly different needs. The new software company, desperate to please every customer to fuel fast growth and maintain high valuations, makes **every** change requested. But with each new customer, the software application becomes that much more unwieldy and harder to maintain and deliver. It's time to grow up the company and introduce a "product roadmap"…

Act IV – The new "product management" department talks to lots of customer and prospects and creates "the roadmap." It shows what will be added to the product by when. To free up the team to create all the great new features on the roadmap, and to stop the source code from getting any more of a mess, the roadmap introduction coincides with the end of every customer getting whatever changes they want when they want them. What makes it on the roadmap comes from careful calculations about which customers the company absolutely wants to keep and which ones it can afford to lose. Phrases like *"but isn't that what I'm paying maintenance for"* start to pop-up in support conversations. The company *might* still make special changes, but only if customers are willing to pay for what they used to get for free. *(Few customers realize this hurts them more than it helps because customizations will likely prevent them from upgrading the core product ever again.)* And companies usually say "no" to those offers of payment once they realize the money can't compensate for the additional code complexity. Red flags of customer dissatisfaction start popping up everywhere…

Act V – Customers re-evaluate their relationship based on the roadmap and the body language they get from the product team. It's clear the software company is in conflict, trying to stay focused on customer desires, but needing to ensure profitability and growth and lay the path for an exit for their investors. Some customers defect for upstart competitors who promise they will become a "marquee" client with the new company adapting its product for their needs (*sound familiar?*). Others resign themselves to a state of mild dissatisfaction and find internal work-arounds for the changes they really want to see in the software package. Some even find themselves "kicked to the curb" if the vendor decides to end-of-life the product and discontinue support.

Act VI – The investor's timeline has hit its end. They "exit," which is most commonly accomplished by a complete sale to another company. The acquirer has other lines of business and re-evaluates the roadmap of the company they just bought. A new direction results, driving more customer defections and increased dissatisfaction from those who remain.

For the general audience, this play was an intriguing drama. For the investors, it was an action-packed musical with lots of fanfare and a story book ending (if the exit was good). But for the original customers, it was a tragedy. Tragic because they started off full of hope, experienced great initial success, but ultimately left disappointed and disillusioned… and likely to repeat the cycle elsewhere.

The key to the customer's discontent was not being able to get features that fit their particular business needs. It was clear from the first act that every customer has needs particular to their own business. But the cost and complexity of writing code to accommodate them all was prohibitive for the software company (unless you're in the list of the top five customers – then you'll get whatever you want).

WHAT ABOUT DEVELOPING CUSTOM SOFTWARE?

Companies who have repeated disappointments with software vendors – or think their needs are truly unique – try building their own software, either through an internal team or a custom coding shop.

Yes, this gets around the "roadmap" issue because now you control it, just like in the days when you were that young software company's only client. But now you have a different set of challenges. Responsibility for managing the interactions between business users and software developers rests with you. Back in the day, "Lost in Translation" was a good movie starring Bill Murray and Scarlett Johansson. Now it's a phrase you'll be hearing often when you ask why your application is behind schedule and you realize your business users didn't explain requirements well to your developers. Next you'll run into the "*d'oh!*" moments. "*D'oh!*" is what Homer Simpson says every time something bad happens to him. It's also what your business users will say when you point out that there's a key capability missing in the software version your developers just spent weeks or months creating.

Issues like these are inevitable in any complex undertaking involving disparate groups such as business users and IT developers. And "lost in translation" and "*d'oh!*" moments aren't the real problem. The real problem is the time it takes to correct them and what that time costs you – both in direct developer costs and lost revenue or the continued high costs that your application was supposed to fix. The cycle time between initial requirements gathering and the first generally available release was long enough. Now it's delayed even more. And what happens if the budget runs out?

I recently met with a business unit of a Fortune 500 company who had paid a developer over $600,000 to create a custom application that now wasn't meeting their needs. They described the internal workarounds they had created to deal with gaps in the application. I asked why they just didn't get the software fixed. They replied that they didn't even consider it as the last time they asked for an enhancement, the quote from the developer (who may have thought they had the company over a barrel) was insanely high.

Companies who go down this path also make a sacrifice without even realizing it. Once the original application is built, they are locked into the technology platform on which it's built. *What happens when the next technology breakthrough comes?* Can they adopt the new capability without a complete rewrite? Even just a few years ago, applications were built expecting users to interact with them only from regular computers. With smartphone use on a continual rise, it's no doubt business users at your organization would prefer a mobile application interface.

WHAT WE DON'T EVEN TALK ABOUT: APPLICATION NEEDS "BELOW THE WATERLINE"

Everything I've talked about so far relates only to processes that are important enough to warrant the investment required for automation. That investment can be substantial. Even with rapidly improving technologies, writing code to create software applications is a laborious process. On top of coders, the price tag for a custom developed solution needs to include quality assurance and support staff. If you're launching a new software company, you'll need to add on the other required functions – sales, marketing, finance, HR, etc. Even a relatively small application can have a high minimum cost to automate. That means automation through traditional software is only likely to make economic sense for mission-critical applications.

But these applications are just the visible part of the iceberg floating on top of the water. Every company has lots of automation needs that just aren't talked about as much since they are smaller in scope with automation benefits unlikely to overcome the minimum cost to create a traditional software application. Most are also processes that occur only within that particular company, making the potential market for an ISV to build a separate application very small.

I get to see what lies "below the waterline" of the iceberg every time I do an operational walkthrough at a prospect or client. The signs are easy to spot. Just look for a big stack of papers on someone's desk. It's usually topped by a printout of a spreadsheet or Access database designed to track all the in-process activity. Of course, the tracking document is always out of date and provides only limited managerial insight. But at least it's some level of automation and the best you could hope for given the cost tradeoffs, *right?*

LETTING GO OF OLD ASSUMPTIONS ABOUT CREATING SOFTWARE APPLICATIONS

If I sound harsh in my commentary about single-purpose applications, I really don't mean it. In fact, let's pause and say a big "thank you" to all of those single-purpose application companies. Collectively, they have unleashed a wave of productivity gains that's certainly fueled the growth of economies around the world, helping to raise everyone's standard of living. Billions of dollars of shareholder wealth have been created in the process, fueling investment gains for everyone.

Single-purpose application companies also deliver value outside of their code. Through their experience with many customer implementations, they've definitely learned what works well and have "best practice" knowledge. You can benefit from this, assuming your business is close enough to the "average" client they serve. Many applications also come with core logic or external data the vendor takes responsibility for updating, such as tax law changes that impact key calculations. It might be worth dealing with the gaps between your core application needs and the software company's roadmap if you don't have to worry about monitoring all those changes.

While these companies deserve our kudos for all they've done, it's time to sunset the model of software development on which they're built. Core technology advances have created new capabilities that stand traditional development assumptions on their head. Let's look at a few of the biggest ones:

Old Assumption #1 – The first assumption to throw out the window has to do with coding. The image that I'm sure just popped into your mind as you read that is of a developer at a computer typing in a language that only a computer could follow. That's how applications were developed. But coding in the latest technology happens visually with users drawing out process flows and pulling from inventories of objects to create required relationships. That doesn't make it "so simple a caveman could do it," but it does move the center of development from an IT group conversant in computer programming languages to the business users who are closest to customers' needs. This can make "lost in translation" issues a thing of the past.

Old Assumption #2 – Another assumption that can go is the idea of manually writing specifications and documentation. If the business users who need the application can create and modify it on the fly, then they don't need to go through the effort of carefully documenting every path of the application so others can build it for them. If the platform they are using is so enabled, it can automatically produce full documentation and even note changes from prior versions, creating a full audit trail.

Old Assumption #3 – The next assumption to fall is that it's cost prohibitive to make frequent changes to applications. With a visual interface and documentation automatically created and kept instantly up-to-date, even small changes are easy to make and don't come with big overhead/setup costs. With lower costs to change, users can adjust their application whenever a new condition arises or a forgotten requirement is unearthed. That's a formula for corporate agility and improved competitiveness.

WELCOME TO THE FUTURE!

So what does this new world look like and what does it mean for companies who develop their own applications as well as independent software vendors (ISVs)?

The market for third-party software solutions has existed because of positive fundamental economics. Until recently, it was usually more cost-effective for a company to license a specialized application from an ISV than it was to build it themselves. Weighing against that was the compromise that the company had to make in not having the application do exactly what they wanted, necessitating technology or process work-arounds. And also, until recently, it was not cost-effective to automate any process that wasn't mission critical.

The technology underpinning software development has been on a steady march to make application creation easier. Advances like the introduction of Platform-as-a-Service (PaaS) removed the complexities of assembling the different pieces required for building applications (e.g. databases, logic containers, presentation layers, and reporting tools). They also separate application development from physical infrastructure limitations by moving everything to the cloud. But to take advantage of PaaS, you still need to be a developer that knows how to write code. That chasm gap between developers and business users has only been recently bridged through the advent of key technologies like Business Process Management (BPM) software. BPM makes development simple enough to shift the center of the work to business users with IT playing a supporting role. This change significantly reduces the cost and cycle time of developing applications and opens up great possibilities for improving organizational agility. It has organizational leadership thinking carefully and asking questions like the following before they license another single-purpose application:

- Why should we license technology that requires us to make compromises to fit the needs of our business when we could easily build our own application?
- Why wait and hope for our needs to show up on a vendor's roadmap when we can own our own roadmap and make changes on the fly, giving us unprecedented agility?
- Why license a multitude of different single-purpose applications with overlapping capabilities when we could recreate just the parts we need as processes run from a single platform at a much lower cost and tailored to our exact requirements?

CLOSING THOUGHTS

It's time to cut your ties with software applications that require acrobatic work-arounds. Set your business users free to develop applications that fit the unique needs of your business. Imagine what can happen if you remove the friction around applications that prevents you from staying focused on your company's growth strategy. The rate of change in business and technology is only increasing. A little bit of paranoia helps, but putting new capabilities into the hands of your team is much more productive. The mantra going forward for businesses that rely on software to automate their processes is "*only the agile survive.*"

ABOUT THE AUTHOR

Evan McDonnell is Appian's Vice President of Solutions. As senior executive at multiple software companies over the past twenty years, Evan has maintained a steady focus on helping organizations achieve competitive advantage and operational efficiencies through better use of technology. Evan continually strives to understand what prevents organizations from utilizing software to its full potential and using that insight to drive breakthrough software business models. Evan has had responsibility for functions including product management, marketing, business development, and mergers and acquisitions. He holds a BS degree in Mechanical Engineering from Carnegie Mellon University and an MBA from Harvard Business School