Build vs. Buy vs. Assemble

contentful

Build vs. Buy vs. Assemble

Which option is best for your business?

Building your own software is not an efficient use of resources if it falls outside of your core competency. It's time consuming, expensive and painful. On the other hand, buying a suite often results in failed promises and paying for more than you need.

Assembling services is the most efficient option. While there is a lot of discussion around "digital experiences," ultimately it's also about changing the way companies build and ship software.

THE FAMILIAR QUESTION: BUILD OR BUY?

So – you need a new solution. Do you build the software yourself, or buy something off the shelf? Traditionally, companies buy pre-built solutions to address a pain or expand beyond their in-house capabilities. On the other end, organizations build software when they need a product that precisely fits their business.

InfoWorld describes the choice to buy as the decision to force your employees to adapt to the software. By contrast, when you build, it's the software that adapts to your organization.

In practice, there's still some build in a buy decision.

Unless the software stands alone (rare), you'll need to integrate it into your other systems and customize it to better fit existing workflows. While software suites often claim infinite customization options, teams are often slowed down by workarounds that don't always function as needed.

"It almost never makes sense to build your own martech software."

– David Rodnitzky, MarTech Today

Growth stifled by build and buy

In a column for MarTechToday, David Rodnitzky wrote that "it almost never makes sense to build your own martech software."

You'll likely recognize the reasoning: You don't have the resources to build, it's not your core competency and software has to be maintained. The speed of innovation also guarantees that whatever you build will quickly become outdated. With so much going against building, why is there even a discussion? Because there are still enormous negatives to buying traditional software suite solutions.

In each case, you're choosing a different way to slow down your organization.

YOU'RE BUYING MORE THAN SOFTWARE

Buying a comprehensive suite means you're also buying a highly opinionated solution. These opinions may fit some use cases, but now you've also bought into them for situations where they don't fit.

For example, a traditional CMS may pre-determine content types. Sure, this is convenient when your marketing team has copy that fits the template. However, as soon as you run into a project that runs counter to the suite's opinions, you're forced to stretch the predefined meaning to accommodate your needs. In other words, your team ends up adapting to the software and you end up letting technology dictate to your business.

In addition to opinions, you're buying features you don't need and will never use – or ones that may actively get in the way of your desired use case. If "draft" and "published" statuses are all you need, you shouldn't be required to push your content through extensive review workflows.

You should be able to organize your process alongside the software, not bend to the software's demands. Forrester weighed in on this phenomenon in its Digital Experience Technology Integration report:

"Many full-featured packaged applications ship with a large core set of features and a prebuilt installer. Unfortunately, these applications hard-code the data and presentation layers, which inhibits the use of existing data or presentation capabilities."

A comprehensive suite solution often overpromises. It's painful to customize, but it's sold as capable of anything. What goes unmentioned is that you often need to adjust your expectations – you're required to agree with the suite's opinions before it will do what you want.

Customers often overbuy, even if they don't need all the additional features. When you add together the customizations and the expanded feature set, projects end up taking a long time to implement. Further, projects often require highly specialized consultants, which add both time and cost.

Unless a suite offers exactly the feature set you need, buying is rarely the most efficient use of time or capital. By comparison, building doesn't look much better.

YOU CAN'T BUILD IT ALL EVERY TIME

Running away from the downsides of buying often means running straight into the wall of building everything yourself. Not very productive.

There are natural resource constraints that keep enterprises from doing this well. For example, even the biggest technology companies don't have unlimited engineers. Like you, they need to decide which projects are core enough to their business to justify building in-house.

As with buying overly complicated suites, building your own software can slow down your team's progress. Shawn Mandel, chief digital officer at TELUS, notes the most important thing you can do is speed up your time to market:

"One of the toughest challenges we see in transformation is getting concept to commercialization right. Getting that thing from proof of concept into the hands of a P&L owner."
- Shawn Mandel

Building from scratch with each project is not sustainable for an organization that wants to move quickly. Buying a ready-made solution has similar downsides. Time to look for a middle ground: individual parts that can be assembled together, as needed, by your digital teams.

"One of the toughest challenges we see in transformation is getting concept to commercialization right. Getting that thing from proof of concept into the hands of a P&L owner."

— Shawn Mandel, chief digital officer, TELUS

Cross-functional digital teams mean faster time to market

The digital revolution has brought about a shift in thinking – and a new set of tools. The most prescient companies are increasing their software development horsepower to focus on their core business opportunities.

They're harnessing a portfolio of cloud-based components to take care of common needs, such as search, email, payments, content, analytics and targeting. When it comes to building the best product, software is no longer only the concern of engineering.

Payments platform Stripe surveyed thousands of enterprise leaders about their software development practices:

- 81 percent of C-level executives think software needs to become more of a core competency for their company in the next 10 years
- 40 percent of developers say they are hindered by custom technology
- 52 percent sat they are held back by legacy systems or technical debt

To meet strategic objectives and remove productivity obstacles, organizations need to move beyond the choice between build or buy.

Assembling reusable tools is a middle ground that helps companies move faster. This move requires an organizational shift to cross-functional teams that connect SaaS and existing APIs – an approach often referred to as a digital factory. Teams have the power to choose and customize their tools can focus on doing what they do best. Assembly lines in the digital factory can then run quickly and in parallel.

With this new approach, customers can "compose their own platforms that they can re-use for many projects [...] Scaling a transformative culture through a digital factory."

Even in traditional organizations, many marketers are interacting more closely with technology. They need to be familiar with staging environments and continuous integration. However, complex suites or heavily customized in-house systems often become a problem for less tech-savvy team members. The assembly approach enables cross functional teams with diverse skills to work together, in parallel.

STACKS, NOT SUITES

Once you've organized around cross-functional teams, you've still only covered half of the formula. If your technology decisions remain the same, then your collaborative team will be as slow as if they were siloed. Rather than building from scratch or outsourcing to a suite, the most successful companies are assembling reusable stacks.

Software stacks are composed of pieces of technology that make up your completed application. Historically, stacks were made up of web servers, programming languages and database choices. Recently they've evolved to include software frameworks, SaaS and APIs.

Elements of a stack can be added or removed depending on the needs of a project. Since each element is hyper-specialized, way less effort is needed compared to the customizations required in a build or buy situation.

Forrester advises companies to move beyond all-in-one solutions-thinking of suites and embrace a systems approach. In this approach, extensions and APIs create flexibility.

> "This modular service philosophy – never building for just one use case – is gaining steam." - Forrester, "Digital Experience Technology Integration: Go Beyond Just A Basket Of Solutions"

By contrast, a suite-based approach locks you into elements pre-determined by the vendor. Before your team can get started on any part of the project, you need to provision, install and configure the suite. All that before you attempt to customize it for your current needs, if that's even possible.

Using stacks allows you to select optimal functionality for each element. You aren't stuck with a one size fits all suite. You can customize workflows without forcing your team adapt to the opinions baked into the software.

Assembling also gives non-engineers opportunities to get involved. While code is often needed to connect elements together, assembly doesn't require the major customization undertaking of suites. Marketers, for example, can use the elements they are most familiar with, and can even get started before the stack is completely assembled.

"The future of innovation is democratizing development," Michel Feaster, CEO and co-founder of Usermind, said on a recent episode of the Andreessen-Horowitz podcast. "It's not having more developers code, it's getting non-technical people [closer to the] code."

One of the ways stacks help democratize development is through standardization. Unlike suites, which often require specialized consultants, the skills to assemble stacks are not unique to a particular software package. The standardized approach allows you to use generalist engineers, or sometimes non-engineers, to integrate multiple elements of the stack.

DIGITAL EXPERIENCE STACK ALLIANCE AIMS AT FASTER ALTERNATIVE

The pieces required to assemble stacks already exist and are rapidly moving towards ubiquity. Linking these solutions together using APIs forms the basis of the Digital Experience Stack. Software built with specific, focused purposes alleviates the pain customers experience when working with vendors that promise to do everything but, in the end, can't deliver consistent quality across the board.

Some of the forward-thinking companies supplying elements of the stack recently created the DXS Alliance to formalize the trend of assembling software stacks.

"The aim of the DXS Alliance is to create a cohesive way for companies to leverage best-of-breed technologies to deliver digital products and the engaging experiences that customers expect."

Among the founding alliance members are Contentful (content infrastructure), Optimizely (experimentation), Atlassian (collaboration), AWS (DevOps/IaaS/PaaS), Tealium (customer data management), Amplitude and Fullstory (analytics), and New Relic (performance monitoring).

Together, the alliance endeavors to help companies accelerate their time to market. Instead of accepting the limitations of a suite solution, they can assemble the best parts of existing software, based on their own unique approach.

ASSEMBLE: THE MIDDLE GROUND BETWEEN BUILD AND BUY

Modern enterprises are familiar with the debate between building and buying software. Reasonable decision-makers know an organization cannot continue by only building or only buying. The common wisdom is to buy in the cases where that makes sense and build when necessary.

The most successful companies, however, are rejecting the either-or question. The happy medium? Assembling elements of a stack.

David Rodnitzky touched on this midway point between extremes in The great martech debate: Build vs. buy. Rodnitzky gives two reasons to build: interoperability and customization.

In each of those cases, you're assembling existing pieces. The elements you choose should make interoperability easier than with any suite. Similarly, customization takes on a new meaning when functionality is already separated into components. You don't need to first tear a suite apart to then put it back together in a different order.

Both building and buying slow down your team. Assembling makes specialized parts available to improve both collaboration and time to market. When you orient your team around a digital experience stack which you can reuse across multiple projects, your teams can use it to create better software.

Reject the false choice of whether to build or buy. Assemble the enterprise modern web stack and accelerate your time to market.