BUILD VS. BUY

WHY GOING IT ALONE DOESN'T ALWAYS WORK IN SOFTWARE DEVELOPMENT

ActiveState

BUILD VS. BUY

WHY GOING IT ALONE DOESN'T ALWAYS WORK IN SOFTWARE DEVELOPMENT

Developers love to develop, and software companies love to build quality software. It's what they do best – or is it? It's possible that some companies can overreach with their abilities and assets, tying up valuable company resources with doomed software experiments. Massively delayed deadlines, exploding budgets, and sub-par results are not unheard of – we see it on the news all the time, sometimes with devastating consequences.

In this paper, we discuss the risks of do-it-yourself software development and make a case for why off-the-shelf or OEM (Original Equipment Manufacturing) may be a better choice for a company's resources, budgets, technology needs, and sanity.

IF YOU BUILD IT

Even the smallest software projects can look deceptively simple to the untrained eye. The most basic of projects still require a knowledgeable team to execute it, hundreds of decisions both big and small, and countless hours spent planning, developing and testing. Any development undertaking will also require a financial investment – companies need to ensure that valuable resources and tightly controlled budgets are correctly allocated. This isn't to say that no software projects should be built in-house, but decision makers must weigh the pros and cons carefully in order to make prudent business decisions designed to help the company compete and thrive.

The software development landscape is littered with stories of DIY setbacks. Even the biggest brands with plenty of in-house expertise and big budgets are susceptible to in-house development problems that can damage a reputation, negatively impact customer loyalty, and most devastating, cost countless dollars. In fact, in a study Gene Kim and Mike Orzen, co-authors of "When IT Fails," conservatively pegged the costs of IT failures, just for S&P 500 companies alone, to be over $100 billion[1] annually.

Here are some notable examples:

HEALTHCARE.GOV WEBSITE COST $834 MILLION

Healthcare.gov may have set the high water mark for the most public IT disaster. The Affordable Care Act (ACA) is President Obama's signature legislative achievement with Healthcare.gov as the driving force to bring it to the public. It's a healthcare marketplace where users can pick and choose between insurance options - but in reality, it is just a simple e-commerce site. It presents users with their options, determine payments and subsidies, collect payments, and submit information to the appropriate health insurance provider. However, despite such a simple premise, problems began surfacing on day one.

From the first minute, the system was overwhelmed with the sheer volume of users. It was later revealed that testing done with only a few hundred concurrent users led to a system crash, the week before launch. There were performance issues, errors, and random bugs that prevented users from completing transactions. On the first

BUILD VS. BUY

day, six people were able to sign up.[2] There were issues with the project from its inception: a flawed bid process, archaic technology choices, and countless last-minute changes which threw off the development and testing schedules. But the most critical error was made in deciding on who would be the systems integrator, a critical role in such a massive project. The government decided it would be best suited to that role:

> *"A former employee of CGI Federal — the private-sector contractor hired to build healthcare.gov — said the government's insistence on being the systems integrator resulted in disastrous consequences for the website."*[3]

In general, the core competency for government agencies is not systems integration or IT and they usually go to external contractors for that reason, but the government wanted control and oversight. Without the deep knowledge and experience required to manage such a large and complex project, it went off the rails pretty quickly. No one had a clear vision of how the different parts were supposed to work together, changes were allowed up until the last minute, and the testing phase was an afterthought.

Due to its high profile nature and its role in the ACA legislation, the website had to go on. It took almost two months for the rescue operations to get the site working and enable users to start signing up. Today the website is functional but the costs were enormous: $834 million[4] plus the job of Health and Human Services Secretary for Kathleen Sebelius.

NHS Scraps £11 Billion Integrated Health Care Records System

The NHS's (National Health Service) National Programme for IT, in the United Kingdom, was said to be the largest civil IT project in history, servicing more than 40,000 doctors and over 300 hospitals[5]. The system was meant to create a centralized electronic health record system, replacing the myriad of systems already in use across the NHS.

The project, with an initial budget of £6.2 billion, missed its first set of deadlines in 2007 and was constantly besieged by delays, contractor walkouts, technical challenges, continuously shifting requirements, and little buyin from users. Four years after missing its initial targets, the UK government announced that they were scrapping the entire project; with final costs double that of the original budget. The Major Projects Authority, set up to review value for money in regards to governmental projects, served the death knell for the project:

> *"There can be no confidence that the programme has delivered or can be delivered as originally conceived. The project has not delivered in line with the original intent as targets on dates, functionality, usage and levels of benefit have been delayed and reduced."*[6]

This project is a prime example of when a business or government ventures into an area where it has no expertise. Government bureaucrats were in charge of a decade long IT project, said to be one of the largest and most complex in history. Its failure was no surprise.

Following the project's cancellation, the government announced that local doctors and hospitals would be allowed to build or buy whichever IT system best suited their needs.

FBI Ditches Custom Software at a Cost of $600 Million

One of the most highly publicized software development project failures belongs to the Federal Bureau of Inves-

tigation (FBI). Before 2000, many of the agency's records were paper-based, preventing collaboration and efficient computer-based analysis. This was one of the major shortcomings that were made public by the events

of 9/11. Prior to those events, the FBI had rolled out plans to develop a virtual case file system that would merge obsolete applications into a single, up-to-date system. This would allow agents to more easily and quickly analyze and share information, helping them further their investigations. However the very public and harsh criticism of the bureau's IT shortcomings following the attacks of 9/11, forced accelerated development timelines for the FBI's new, custom-built system.

When the project was finally delivered, significantly over its $170 million budget and behind schedule, the case management software was considered unusable and shelved. The project's failure was attributed to multiple factors: poor system architecture, lack of IT project management expertise, unclear and shifting requirements, and time-pressured project schedules. Another critical mistake was trying to build each component from scratch as noted in the Washington Post:

> *"Along the way, the FBI made a fateful choice: It wanted SAIC [the contractor] to build the new software system from scratch rather than modifying commercially available, off-the-shelf software."* [7]

A month after the project was shelved, the FBI announced it would be building a new system based on mostly offthe-shelf software rather than re-attempt a custom job.

IT REALLY IS THAT HARD

These high-profile projects demonstrate that flawless custom software is difficult to achieve, even for large, wellfunded organizations. A CHAOS Summary from the Standish Group reported that in 2015, 71% of software projects did not meet their development goals,– 52% fell short on features, went over budget and were delivered late, and an additional 19% either weren't delivered at all or were never used.[8] According to Standish Group, these figures are on the rise and demonstrate just how difficult it is to deliver top-quality software.

HIGH RISK, LITTLE REWARD

As indicated in the examples above, a variety of complications can occur when teams set out to build complicated software solutions, in-house. Most are optimistic about what they can accomplish and plans are often ambitious, pushing development teams beyond their core competencies — what they're actually really good at. Here are some things to consider and determine whether your project is high risk with little reward:

In-house expertise

Does your organization have the necessary skills and expertise to produce to the level of quality needed? If the answer is no, it can result in a subpar product that lacks the required features and functionality. Outside experts can be brought in but this will heavily impact project costing and delivery timelines.

Budget constraints

Many, if not most, companies do not accurately ascertain the level of financial commitment required to complete a high quality software project. There can be nasty surprises that pop up along the way and result in projects costing much more than originally anticipated. In worst-case scenarios, budgets that

BUILD VS. BUY

have gotten so far off-track that the companies cannot financially recover. Also, budgets and resource allocation should, but rarely do, take ongoing maintenance and support into account.

Time constraints

In general, software projects are started because there is an urgent business need. Taking the time needed to plan, build, and test it may not be possible given tight deadlines.. It may sound great to have a custom tailored solution, but organizations risk giving up their competitive advantage as they lose critical time waiting for the custom solution to be implemented. In today's market it's not necessarily the most featurepacked solution that wins, but the one that gets there first and simply gets the job done with the least barriers to adoption.

Off-the-shelf solutions

If there are off-the-shelf solutions available, with all the necessary features, there really is no reason to take on the headache of building. A pre-built solution will have been tested over time, by multiple users, and have the necessary support available to address a company's needs and issues. Also, an off-the-shelf solution means that organizations can get back to doing what they do best – faster.

Competitive advantage

Companies should only venture to build software that is a distinct competitive advantage for them and a core competency. If neither of these factors is true, then they should not spend valuable time and resources on it. It will only result in a less than optimal solution that will not best serve the business. Opportunity cost is real and companies should focus on what they do best.

When enterprises don't have the know-how, budgets, or time to build it themselves, it might be time to turn to OEM software solutions. The next section covers dynamic programming languages (sometimes referred to as scripting languages). Their usage is widespread (and growing) in enterprise and government software projects, due to their efficiency, ease-of-use, and how they can accelerate the development process.

DYNAMIC PROGRAMMING LANGUAGES: BUILD OR PRE-BUILT

Millions of developers use dynamic languages to solve problems with building and integrating heterogeneous systems. Whether it's manipulating data, working with devices, or prototyping, dynamic languages (such as Perl, Python, and Tcl) are popular because they save development time, improve the user experience, and provide users with flexible scripting options.

Developers often custom assemble dynamic languages because they assume they're easy to install, test, and maintain. However, they tend to underestimate the work involved in integrating and maintaining dynamic language distributions because they are so ubiquitous. More than 97% of Fortune 1000 companies use dynamic languages to power various applications and accomplish daily IT tasks.

Perl scripts are commonly used to run complex computational and integration tasks. Python's support for basic Internet protocols makes it a natural fit for web applications. It is also becoming increasingly popular for heavy datacentric scientific computing and financial modeling

BUILD VS. BUY

applications. Tcl is ideal for rapid prototyping, GUIs, automated testing, and is widely used in Electronic Design Applications (EDA), Field-Programmable Gate Arrays (FPGA), circuits, and semiconductors.. Dynamic programming languages – Perl, Python and Tcl in particular – are efficient, easy-touse, quick to learn, and can be used for a wide variety of different software tasks.

Another big reason that developers tend to deploy Perl, Python, and Tcl themselves is because they are open source. They are widely available, with no licensing fees, and no need for purchase orders. Developers can simply download the code and get started. However, there are big drawbacks to using open source versions of these dynamic languages:

Technical Support

Open source versions of Perl, Python, and Tcl don't require software licenses but that also means that there are no service level agreements for support either. Whenever there are issues, there is no one contractually obligated to assist a company. Instead users have to rely on developer communities who may or may not answer questions on web forums, mailing lists, and ad hoc support databases. Even worse, users may end up with a half dozen conflicting responses when they do. No access to reliable tech support can be a major setback when a team lacks in-house dynamic language expertise and is still trying to meet release dates and produce quality software.

Maintenance and Upkeep

Installing and maintaining an open source dynamic language is an ongoing task. Developers must continually maintain and update open source code to ensure that bug fixes and feature upgrades are up-to-date. These kinds of tasks are not top of mind for most developers and can easily fall to the bottom of a busy developers' task list, exposing their installation to security risks and can lead to product degradation over time.

Licensing

Open source software is almost all covered by one or more licenses that must be adhered to. Some are fairly permissive, some much less so, and may require any changes to the software to be published if the modified version is distributed. Furthermore, specific modules within an open source distribution may have their own license terms, which can be considerably stronger than the basic language license. The time and effort spent managing licenses can easily grow exponentially, and the cost of even an inadvertent license violation can be considerable, as it may involve intellectual property lawsuits, loss of reputation, and even the inability to legally distribute your product until the problematic open source component is removed.

These shortcomings make open source distributions of dynamic languages an ideal off-the-shelf option for development teams that want to get the most out of Perl, Python, and Tcl without shouldering the responsibility of ongoing installation, maintenance, and licensing.

BUILT BY ACTIVESTATE: ACTIVEPERL, ACTIVEPYTHON, AND ACTIVETCL

At ActiveState, dynamic languages are a core competency, and have been since the late '90s. These language

distributions are considered the industry standard and are used by millions of developers around the world.

BUILD VS. BUY

They come pre-compiled for out-of-the-box installation and include core binaries, popular modules, and complete documentation. Whether they are being used for business- or mission-critical applications or open source projects, ActiveState distributions save development time, help get products to market faster, and improve the user experience, both for development teams and customers. Here's how:

> *"Our story at Numara is that FootPrints makes it significantly faster, easier and more cost effective to run a support environment than it would be to build your own solution in-house. I thought we should practice what we preach and let ActiveState do the same for us."*
>
> -MICHAEL KRIEGER, NUMARA SOFTWARE

Faster Software Development

Dynamic languages often enable key functionality and must be considered early in the development cycle. ActiveState language distributions are pre-compiled, aggressively tested, and work across numerous platforms. They provide a solid foundation so that developers can immediately begin coding with confidence, knowing that a project cornerstone is in place. By choosing an enterprise-grade language distribution rather than assembling one in-house, companies will save time on installing scripts, running test cycles, and troubleshooting cross-platform issues throughout the development lifecycle.

ActiveState OEM customers get direct access to the world's best Perl, Python, and Tcl engineers. Development issues are resolved privately, not in a public forum and enterprises will get unlimited incidents, troubleshooting, emergency in-production coverage, and guaranteed fast response times and fixes. When organizations are under pressure to meet deadlines and get products to market, priority access to ActiveState's experts will keep a project on track.

> *"Not only is it reliable, scalable and enterprise-ready, ActivePerl saves CA development time and dollars."*
>
> - LAWRENCE BACKMAN, VICE PRESIDENT, CA

Better Quality Software

Enterprises reduce the risk of project failure if each component in a software program functions at its very best. ActiveState OEM distributions for Perl, Python, and Tcl are guaranteed to be the highest-quality language distributions available, rigorously tested for security, stability, and quality. OEM products also include regular updates and fixes so that companies are always working with the latest, most secure release. Working with high-end dynamic language distributions enables the development of higher quality products.

Cross Platform Product Support Out-of-the-Box ActiveState language distributions are available on multiple platforms including, Linux, Windows, Mac OS X, Solaris, AIX, and HP-UX. There is no need to spend time preparing dynamic languages for cross-platform deployment or troubleshooting common platform issues. ActiveState distributions operate to the same level of quality across all these diverse systems promising a consistent user experience.

> *"The OEM Agreement allows us to bundle Perl with our product so the installation and maintenance process becomes easier and controlled. We estimate that the [ActivePerl OEM license] saves us about 50K Euro per year on support and possible missed deals."*
>
> -WALTER VERHOEVEN, CREATIVE ASSOCIATES

BUILD VS. BUY

Hassle-Free Redistribution Rights

If Perl, Python, or Tcl must be redistributed with a software product then the organization is responsible for the appropriate licensing. However, this can be a complex undertaking since there are dozens of different open source licenses. Open source languages are made up of thousands of libraries, modules, packages, and frameworks — each with unique licensing requirements. Some open source licenses, such as the common GPL license, require companies to contribute all code back to the community including their own modifications. This can be problematic for companies that want to use dynamic languages,but still keep their IP private.

Not doing due diligence when it comes to licensing requirements can cost a company in intellectual property infringement lawsuits, hefty lawyers' bills, and public embarrassment. ActiveState OEM licensing eliminates administrative overhead and potential legal risk that can occur when including open source software in commercial products. ActiveState carefully reviews all open source licenses that are a part of ActivePerl, ActivePython, and ActiveTcl and provide warranties for redistributing the code safely. When ActiveState handles the licensing, organizations can rest easy knowing that they are safe from legal risk and IP infringement. Plus, they can spend time building software, not wading through complex licensing requirements.

Increase Customer Satisfaction

Bundling ActivePerl, ActivePython or ActiveTcl with a product improves your clients' product experience since they don't have to download the code separately. The result is a seamless out-of-the-box experience for customers where the application works with fewer external requirements. There's also no risk that customers will erroneously download incompatible versions of the software.

An organization demonstrates reliability, integrity, and professionalism by licensing open source correctly. If customers know their purchase is covered by open source licensing with ActiveState, they don't have to go to the hassle and expense of purchasing additional indemnification or support.

Finally, if customers have specialized requirements for dynamic languages, ActiveState can help with custom development such as maintaining abandoned modules, auditing code, and extending tooling. ActiveState customers can add Perl, Python, and Tcl experts to their team at a moment's notice.

BUILD VS. BUY

QUALITY COMPONENTS FOR QUALITY SOFTWARE

Dynamic languages are just one factor in a complex software project. However, when a company assembles software from top-quality components, such as ActiveState language distributions, they will start with a solid foundation from which to construct a first-rate software system that customers will love.

For help with dynamic language development, management, and licensing, please contact ActiveState at business-solutions@activestate.com or 1.866.510.2914 (toll free in North America) to connect with an ActiveState dynamic languages expert about your upcoming development projects.

1 Worldwide Costs of IT Failure - http://www.zdnet.com/article/worldwide-cost-of-it-failure-revisited-3-trillion/

2 Where did Healthcare.gov Go Wrong? http://blog.smartbear.com/code-review/where-did-healthcare-gov-go-wrong-lets-start-witheverywhere/

3 Troubled Obamacare website wasn't tested until a week before launch - http://www.washingtonexaminer.com/troubled-obamacarewebsite-wasnt-tested-until-a-week-before-launch/article/2537381

4 Obamacare Website Costs Exceed $2 Billion, Study Finds - http://www.bloomberg.com/news/articles/2014-09-24/obamacare-website-costs-exceed-2-billion-study-finds

5 The World's Biggest Civilian IT Project Finally Looks to Have Failed But Is the NHS IT Failure a Surprise - http://www.computerweekly.com/blogs/outsourcing/2011/09/the-worlds-biggest-civilian-it-project-finally-looks-to-have-failed-but-it-is-no-surprise.html

6 Government to Scrap NPfIT IT Programme Today - http://www.cio.co.uk/news/outsourcing/government-to-scrap-npfit-nhs-itprogramme-today/

7 Dan Eggen and Griff Witte, "The FBI's Upgrade That Wasn't," The Washington Post, August 18, 2006, http://www.washingtonpost.com/wp-dyn/content/article/2006/08/17/AR2006081701485.html

8 Standish Group 2015 Chaos Report - http://www.infoq.com/articles/standish-chaos-2015

ActiveState Software Inc.
sales@activestate.com

Phone: **+1.778.786.1100**
Fax: **+1.778.786.1133**

Toll-free in North America:
1.866.631.4581

ActiveState®

ABOUT ACTIVESTATE

ActiveState believes that enterprises gain a competitive advantage when they are able to quickly create, deploy and efficiently manage software solutions that immediately create business value, but they face many challenges that prevent them from doing so. The company is uniquely positioned to help address these challenges through our experience with enterprises, people and technology. ActiveState is proven for the enterprise: more than two million developers and 97 percent of Fortune 1000 companies use ActiveState's end-to-end solutions to develop, distribute, and manage their software applications written in Java, Perl, Python, Node.js, PHP, Tcl and other dynamic languages. Global customers like Cisco, CA, HP, Bank of America, Siemens and Lockheed Martin trust ActiveState to save time, save money, minimize risk, ensure compliance and reduce time to market.