AN

EXPERT'S GUIDE TO BUYING SOFTWARE

Learn the factors that really matter when you're investing in software

By Frank Defesche

Table of Contents

1

Are all software companies the same?

After 20 years working in enterprise software, I thought I had it all figured out. Over the last year though, I've realized that software has changed but the evaluation process hasn't. I was speaking with a division of a Fortune 100 company seeking to improve their manufacturing processes through new software. When they showed me their software comparison matrix, every vendor checked the same boxes. *What?*

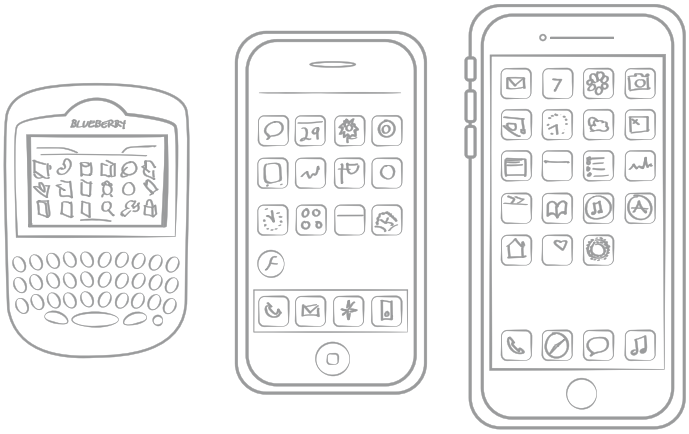
Is All Software the Same?

Feels like it. So, if we all check the same boxes why do business outcomes vary so widely? What is really driving value and success for a business?

Beyond the Matrix: Today's Savvier Software Buyers

If you think about it, software constantly evolves. So, a software feature comparison today is inaccurate typically by the time a company has it rolled out.

Do phones have the same features today that they did 15 years ago? Or even 15 months ago?



Clearly not.

Given feature matrices limited value, it's no surprise that a vendor's feature set is receding as the primary method that companies use to select software. Some buyers are getting smarter; they are looking behind the curtain and shifting their evaluation focus to the actual software delivery method:

- Do I install the software on-site?
- Does the vendor host it?
- Is it multi-tenant cloud or single-tenant/private cloud?

Oh, and there's one last question: *What the hell does any of this really mean?*

What Really Matters?

The heart of the answer is flexibility. Is the software agile enough to keep pace with the accelerating demands of the business and the market it serves?

A focus on flexibility can be the beacon to guide companies through the noise of rotating software buzzwords. And contradicting rhetoric around them.

I've had the great fortune to have a front row seat during the birth of Software as a Service (SaaS or Cloud) as an early member of salesforce.com. I supported hundreds of enterprise deployments and software evaluations during my long tenure there, and for the past decade at Veeva Systems. I've watched first hand as companies achieve long term success and moving a lot faster than they ever thought they could using cloud software.

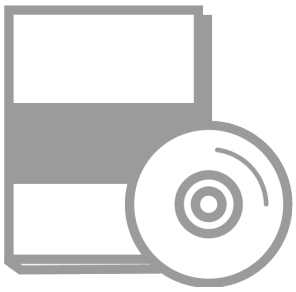
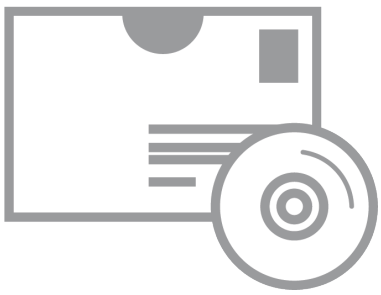
Given the amount of mystery and misunderstanding about cloud software, I feel compelled to share my experience to uncover what really matters when making software investments. Over the next few weeks we'll be releasing posts that look at this question from several angles to equip software buyers with the context and questions they need to select truly great software.

Because truth be told, we're not the same...not even close!

2 Defining Software Models

Before we plunge into business-impacting aspects of software like security, cost, risk, innovation and flexibility, we need to understand how different types of software are accessed, deployed, modified, and managed. This is often referred to as “software delivery models.”

There are three main software delivery models: on-premise, private cloud, and multi-tenant cloud. The differences between them can have immediate and profound ongoing impacts on your business, more than you might think. They are as different as buying DVDs, renting DVDs, or streaming movies on demand.

On-Premise	Private Cloud	Multi-Tenant Cloud
		
Buying a DVD Static, expensive, limited future	**Renting a DVD** Same on-premise technology now hosted	**Streaming** Constantly innovating, always current, no up-keep

On-premise

On-premise is what its name suggests – software hosted and maintained on your company's premises by your IT department. In our analogy, it's akin to buying a DVD.

The on-premise delivery model dominated the software era of the 1980's and 1990's. It was born and flourished in a time when the internet was still in its infancy.

With on-premise software, the customer owns a perpetual license of the software as it is installed at their facility. As the company changes and grows, it is fully incumbent upon IT resources to maintain and evolve the software, keep hardware refreshed, and pay the vendor for software upgrades and maintenance. Cost and effort increases overtime and become very unpredictable. On-premise software is, afterall, a depreciating asset.

Private (or Single-tenant) Cloud

The private cloud delivery model still provides each customer with their own instance of software. The key difference compared to on-premise software is that with private cloud the software vendor hosts the supporting infrastructure in its data center or on a cloud infrastructure (ex. Amazon Web Services, Microsoft Azure).

Despite appearances, private cloud is closer to on-premise than it is to multi-tenant cloud. In many cases, private cloud providers are merely disguising on-premise software with a cloud delivery model. It is like getting DVDs in the mail through a subscription service instead of buying them.

By hosting the supporting infrastructure, this private cloud delivery model shifts the responsibility for security, application access, operations, general maintenance and accountability from the customer's IT organization to the vendor, which is an advantage over on-premise. The software vendor can also deliver application updates, ensure performance, and manage security updates. However, each instance of the software is still siloed, which makes it subject to the same inefficiencies and risks as on-premise.

Multi-tenant Cloud

Multi-tenant cloud (or Software as a Service (SaaS), or pure cloud) is the software delivery model where multiple customers leverage a single version of software and co-exist on a shared infrastructure that is managed and hosted by the software vendor.

Much like systems used to separate bank accounts or medical records, multi-tenant cloud vendors

use virtual partitioning (think firewalls) to securely segregate customer's data, configurations, and customizations.

Each customer is on the same, most current version of the software, but each customer's configuration or customizations are their own. In other words, each customer (a.k.a. tenant) shares computing resources, but each "tenant" is autonomous and remains invisible to other tenants.

The multi-tenant software delivery method has become the preferred delivery method for the software industry, and for good reason. The majority of today's leading software companies deliver their products via a multi-tenant architecture, and nearly every new software company started in the last 10 years has had a multi-tenant cloud strategy.

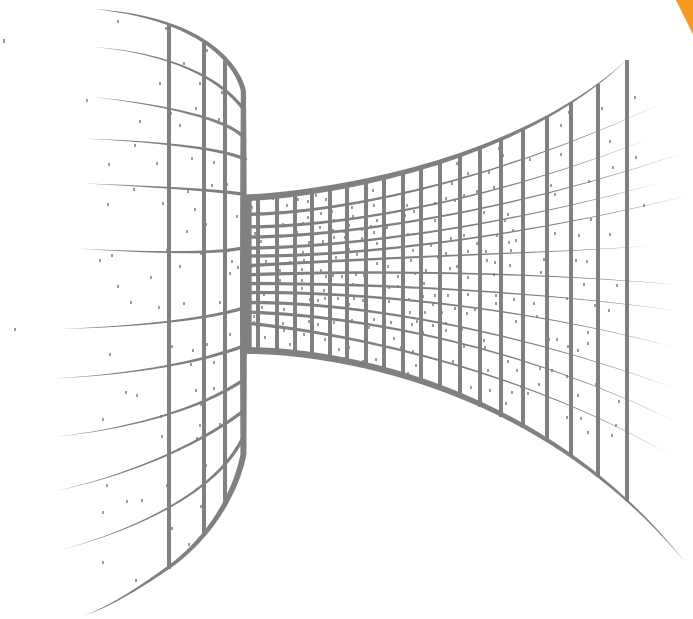
The reason is clear; maintaining only one version of code allows multi-tenant software vendors to innovate faster and provide more value to all their customers with every investment. This is why people often say that multi-tenant cloud has democratized the software world by providing the same value to both small and large companies. Multi-tenant cloud software is an appreciating asset.

The multi-tenant infrastructure also benefits from a crowdsourced model for quality. If there is a defect, it's found faster. When fixed, it is fixed for all customers. Every day, every user is "testing" the same system. The benefits of one version of software and a multi-tenant infrastructure pave way for significant gains in security and scalability. We will talk about that next time.

Security and Scalability

3

Security and Scale with On-Premise



Security and Scale in the "Cloud"

The biggest area of confusion for software buyers is security and scalability. Sadly, this confusion has been fueled by software vendors that are not multi-tenant cloud. They exploit common misconceptions and paranoia in most buyers. After all, we're not all software engineers.

Their main strategy is to suggest to buyers that there is greater risk with a shared infrastructure. That with multi-tenant cloud software, data can get exposed to others, either inadvertently by a bug or maliciously by hackers.

In reality, multi-tenant providers know that security and scalability are table stakes for success. Their customers include governments, banks, pharmaceutical companies, and other industries that need to protect highly sensitive information. If a software vendor has security or scale issues, they are done!

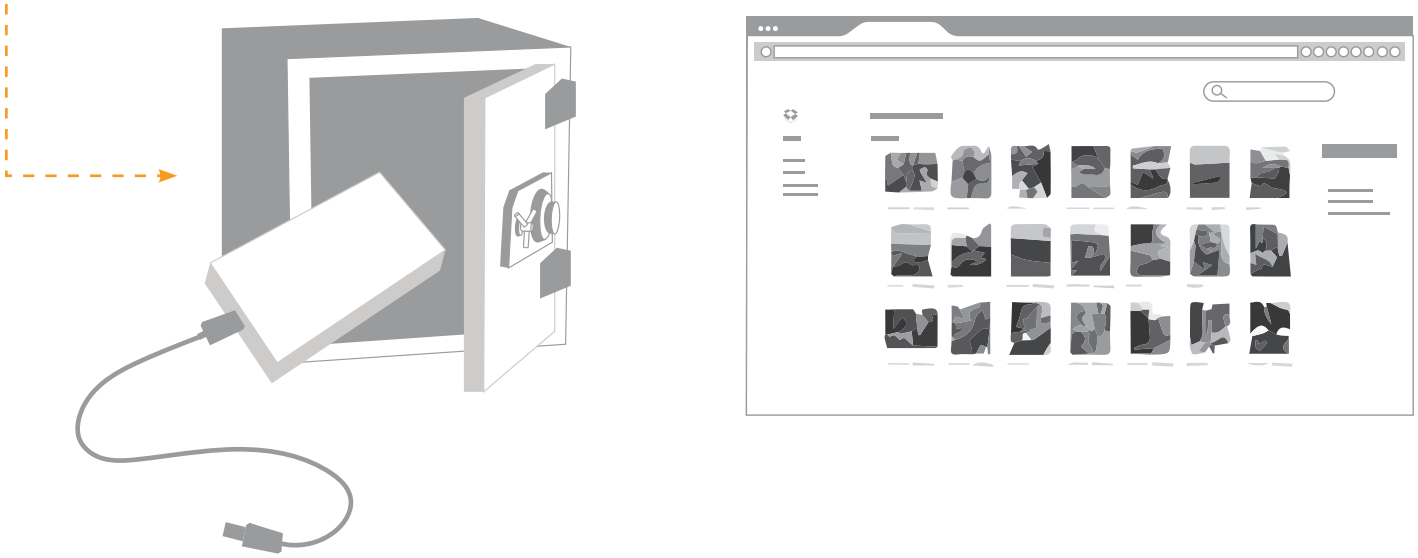
For this reason, multi-tenant software vendors apply greater expertise, investments, and resources to ensure the infrastructure and their customer's data is as secure as possible. The more established the multi-tenant software vendor is, the more investments they make in security and scalability. This makes multi-tenant software the most secure option.

Private cloud vendors are mostly on-premise vendors that are moving to the cloud. Private cloud is an easier strategy for them as it requires less knowledge and provides the ability to leverage their outdated software. By comparison multi-tenant cloud vendors were born in the cloud era so they tend to have greater expertise, techniques, and tools around all the layers of cloud security.

The Cost of Security

Security is expensive. It requires constant investment to stay ahead of new vulnerabilities. When the first iPhone came out, it didn't have many security features. No remote wipe. No fingerprint identification. These are common security features today but they are the outcome of many investments and innovations Apple has made over time. The same concept can be applied to securing enterprise software.

Security of on-premise software is like keeping your family photos on an external hard drive. To make it more secure, would you buy a safe and put your hard drive in it? Unless you are my long-retired mother, answer is probably no. Security of private cloud and multi-tenant cloud is like putting family photos in iCloud or DropBox.



On-Premise: Since your data is managed on your own company's servers, your company needs to make ongoing investments to keep it safe. The burden of continuous improvement in security is not something every company has the ability to take on. Most companies are focused on growing their business, not software security.

Even many of the world's best software and network security companies lean towards cloud vs. on-premise.

When all your company's crown jewels and IP are on-premise, the risk exposure is higher. If you break through the company's firewall and have access to one service, you have easy unfettered access to all the crown jewels. Protecting data, assets and IP in this model is more expensive than a cloud based model.

Private-Cloud: The cost of security is passed to the vendor, which is a good thing. But since every customer uses a separate infrastructure, the cost to improve security is not as efficient because it has to be managed and upgraded instance by instance. This can deter a private cloud vendor from making security investments putting you at risk.

Multi-tenant cloud: The vendor is accountable for security. Again, this is a good thing. The difference is that the economies of scale for security investments for a multi-tenant infrastructure are greater than private cloud. Every investment benefits every customer automatically. Greater ROI leads to larger and more frequent investments. So when new innovations in security become available, your software vendor is more inclined to make those investments and you get the benefit.

Data security is not the only area where the multi-tenant model works to your benefit. The software vendor has greater visibility to how customers are using the software which improves quality control. If the software has a defect, it is detected and fixed before it impacts your business. This makes the vendor smarter and more proactive.

Scalability

Software should be fast, regardless of where you are globally or what device you are using. As any business and its related information expands, application performance is an imperative. Take for example the first-generation iPhones, which had options of 4GBs and 8GBs. Perfectly appropriate for consumer needs when they were released. But with the rise in image fidelity, social media and "there's an app for that" generation, a 4GB or 8GB iPhone would not be usable today. Not to mention the early iPhones did not have the processing power to support the current version of IOS. Now take that concept and apply it to mission critical enterprise software for your organization.

Simply put, scalability comes down to processing power, storage, and business-mandated performance standards.

On-premise: The burden of scale and performance is 100% on your company's IT department. More data, more servers. More users, more servers. More servers, more cost and resources to maintain

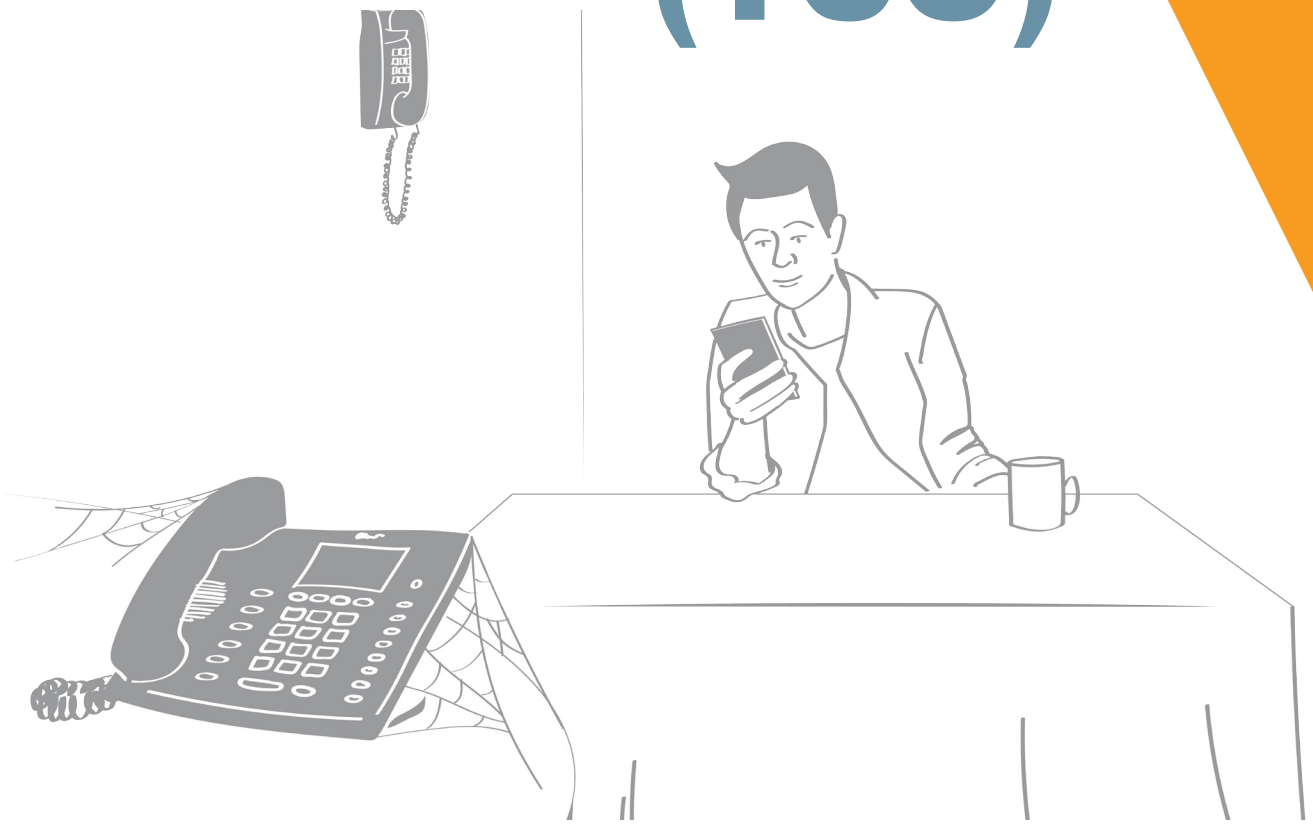
them. IT organizations are getting squeezed. IT needs to innovate to support the business more than ever - will they invest in innovation or be forced to use their resources to grow a server farm?

Private cloud: Because the cloud software is on a dedicated infrastructure, if your dataset or application requires more storage and processing power, you may incur additional costs. In effect, your cost reflects peak level activity and does not enjoy economies of scale from a shared environment like multi-tenant cloud.

Multi-tenant cloud: Similar to security, investments go a longer way in a multi-tenant architecture. In a frictionless manner with minimal to no disruption to end users, multi-tenant software vendors build a service that scales to meet the aggregate demands of all their customers. Multi-tenant vendors build and continue to improve an infrastructure elastic enough for all their combined customers. The result is better performance, no surprise or hidden fees, and less disruption to people getting their jobs done.

Total Cost of Ownership (TCO)

4



I'll admit it: I still have a landline phone at home.

Two in fact - a relic AT&T phone with a voicemail tape recorder that has hung on the wall for 20 years, and another that came as part of a "triple play" bundle from my cable provider. I have never used this second phone line, but it somehow makes my cable package cheaper. Don't ask.

And I have my iPhone with Verizon wireless, which is how I make 95% of my calls.

Three different contracts with three different providers that each charge me for phone service. It's not only horribly redundant, it's exhausting! All I want is a good service at a good price.

In many ways my phone dilemma mirrors the challenges facing today's software buyers.

Vendors tout features that confuse buyers rather than differentiate. Pricing models are so complex that you need a calculus degree to compare them. Sure they all claim to have the best technology, but in the end all I need is to make a simple phone call.

Seriously - after I check the box “can reliably make a phone call”, all I care about is PRICE!

Establishing a Common Cost Framework

Cost remains the undisputed heavyweight champion when making software investment decisions. Yet uncovering the true costs of on-premise, private cloud, and multi-tenant cloud software requires the buyer to be part savant.

Different software delivery models shift costs between licenses, maintenance, upgrades, resources, and other areas. This lack of parallelism makes it difficult for buyers to establish accurate estimates for total cost of ownership.

While it’s challenging to align vastly different software models to a common cost estimation framework, it is possible. Here’s how.

First we must understand where different costs originate. Aligning all costs to categories helps create an honest picture spanning the different delivery models. Three common categories are:

- Upfront license costs
- Ongoing direct and indirect costs
- Cost predictability

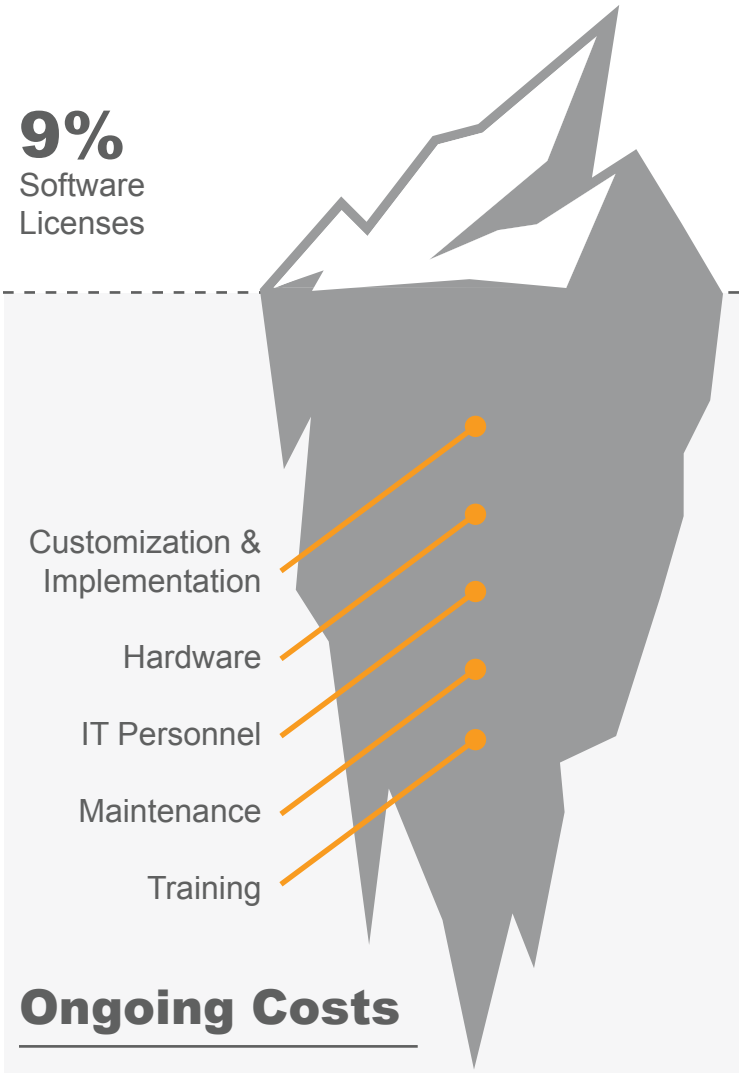
Cost predictability, although not often considered, is an important attribute so I’m forcing it into the discussion. There is also value (or ROI) differences that come into play but, for simplicity’s sake, let’s limit our scope to cost.

The Shell Game of Cost

Here’s the good news. All enterprise software companies charge for their software. Whether it’s called a perpetual license or a subscription service, it’s simply a fee for the legal right to use a vendor’s software. But buyer beware. Underneath the shell, you might find a laundry list of hidden expenses that contribute to the overall cost of ownership.

The image below is a classic representation of this:

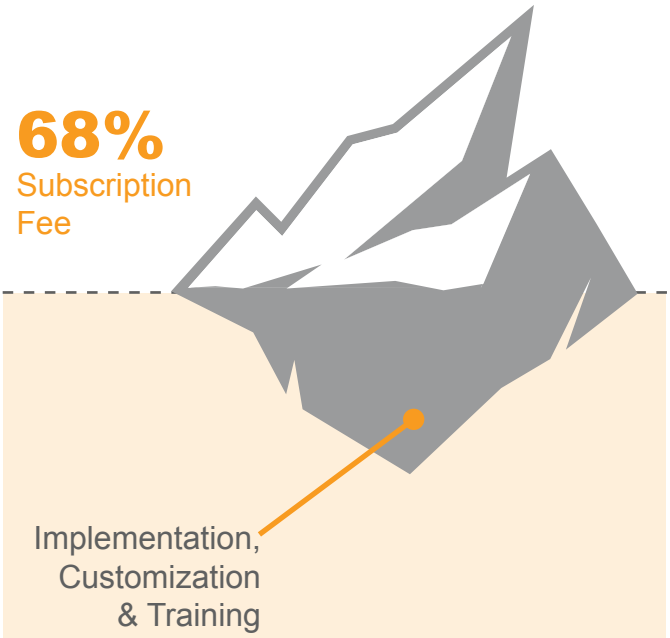
On-Premises



Ongoing Costs

- Apply Fixes, Patches, Upgrade
- Downtime
- Performance tuning
- Rewrite customizations
- Rewrite integrations
- Ongoing burden in IT
- Maintain/upgrade hardware
- Maintain/upgrade network
- Maintain/upgrade security
- Maintain/upgrade database

Cloud Computing



Ongoing Costs

- Subscription fee

The cost model for on-premise software reminds me why I will never accept a free cat. Vet bills, litter box, cat food, and the cost of shredded furniture add up fast and continue for years. As a buyer, a common cost framework helps you tease out layers of costs to produce an honest cost comparison.

Normalizing The Cost Equation

I'm gonna go out on a limb and claim that CIOs and CFOs do not like surprise costs. Here are six factors to consider that can dramatically impact the total cost of software:

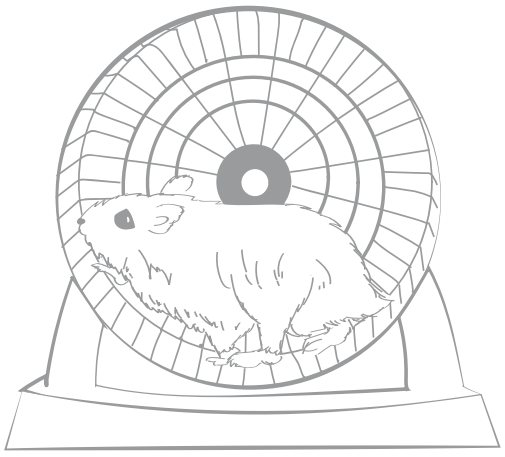
1. Compatibility: Every year or two, Microsoft releases a new version of Internet Explorer and Google releases a new version of Chrome. Your software needs to be able to work on it. With on-premise cloud, 9 times out of 10 upgrading to a new browser will incur cost - likely double if your company uses both PCs and Macs. With multi-tenant cloud, the vendor ensures compatibility as part of the license fee. With private cloud, it depends, so ask for details as applicable.

2. Upgrades: With on-premise software you are purchasing a specific version of the software. Think of a computer purchased in 2000 that had Microsoft Windows 2000 and Microsoft Office 2000 on it. Every few years new versions of both the Windows operating system and Office applications came out.

Fast forward to today and you've blown past dozens of browser and software releases so you'll have a huge upgrade bill. Extrapolate that cost across hundreds or thousands of computers. That is how companies get trapped into paying for expensive upgrades every few years. Or worse, staying on an older version and crossing fingers that your aging version stays supported by the vendor and remains compatible with operating systems and browsers.

With multi-tenant cloud software, every customer is always on the same version of the software, the current one. We like to say this makes the software future proof.

3. Elastic Computing: Say your company grows and the usage of your enterprise software begins to climb - what do you do? If you have multi-tenant cloud software, IT can hit the snooze button in the morning because scaling up or down is nearly painless. If your company is still maintaining on-premise software, IT is likely sleepless and worrying about hardware, storage, networking, cooling systems, and space to grow your data center.



Stop running on the hamster wheel. Computing is cheap and should be elastic to accommodate growth and occasional usage spikes.

The elasticity cost variable is, perhaps, the most critical to normalize when assessing the total cost of ownership. For instance, let's say on Monday mornings you have an application usage spike. Well, for on-premise software, you need to have sufficient capacity in place to handle the surge. This fact forces companies to purchase and deploy infrastructure for "what if" scenarios on usage.

If you have a private cloud solution, this "what if" scenario is likely hidden in a concurrent user license fee, where companies have to purchase sufficient licenses to handle the maximum number of users accessing the system at a given time.

Not so for multi-tenant cloud - it's elastic. Companies subscribe to a service which transparently expands to handle spikes in usage, without inflicting additional cost or complexity to the customer.

4. Modern User Experience: Another murky cost consideration is the user experience. A great user experience fosters adoption, reduces training costs, and promotes data accuracy. A poor user experience destroys any ROI model.

What makes a good user experience? They say that beauty is in the eye of the beholder, and that's partially true here. Software designed 30 years ago has not embraced the modern, intuitive web design templates that make shopping on Amazon so easy.

Don't be shy about looking behind the curtain: ask the age of the system that you're evaluating, count the clicks to perform common tasks, and see if you can perform a task without a manual or training. I mean, have you ever needed a training class for a mobile app?

5. Pace of Innovation: On-premise solutions proudly tout their annual release cycles, while multi-tenant cloud vendors are pushing out weekly updates. Users' access to innovation couldn't be more different.

Yes, this is a challenging cost metric to factor into the buying model. But if agility and innovation are key business drivers, then you will need to account for the relative pace of innovation of software deployment models.

6. Future Proof: We touched briefly on the notion of future proof earlier, but it goes far beyond upgrades. For example: If you were to purchase Documentum or Sharepoint today and deploy either system in a mission critical environment, you would be stuck. There is ZERO R&D investment into either system. None.

While Microsoft is planning a full rewrite of Sharepoint to support its new cloud environment, Azure, this will require current Sharepoint customers to perform a full re-implementation.

So, before you buy, make sure that you glance at the horizon and ensure that the product that you are selecting will be around for awhile. On-premise solutions have no future and that needs to be factored into your cost equation. And while you're at it, you should ask the same question for a single tenant, private cloud offering as well. Most of those systems are being sunsetted in favor of the multi-tenant cloud delivery model.

On-Going Costs

Clearly peeling back the layers of the onion helps a buyer to normalize software costs. It's always helpful to pull the hidden fees into view. But what about on-going cost?

Here's my recommendation: put your software deployment scenarios into a three to five year cost equation and be sure to include the sustainment costs of all the hidden fees. Then derive the net present value across all three options.

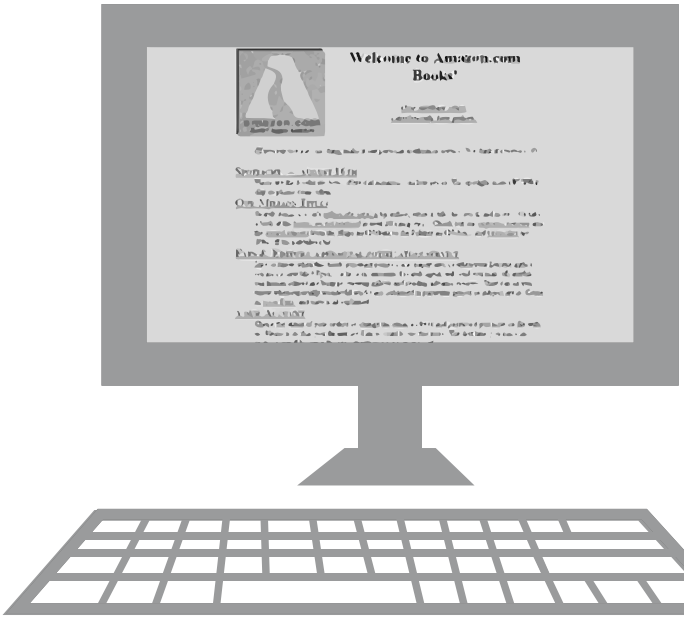
But before you finish, consider the multiplicative property of ZERO. If the user experience is poor, then the software's value is ZERO. If there are no future R&D on the application or underlying platform, then its value is ZERO.

So, do your research, try to normalize costs as best you can, and make today's decision with eye towards tomorrow to help protect your investment.

As for me and my phone dilemma? I'm consolidating behind Verizon. The user experience is superior, I always have my iPhone with me and I use over 30+ mobile apps. Lastly, it provides the easiest way for me to stay in touch with my family...texting! So, yeah, Verizon is cheaper, easier, and let's me keep pace with the world around me.

Innovation 5

Innovation is a loaded and grossly overused term. To put “innovation” into context, let’s take a stroll back to Amazon circa 1996 compared to today’s Amazon.



Amazon circa 1996

Innovation, right? Quite a remarkable difference in just 20 years. How many upgrades or training classes did you take as Amazon innovated and expanded its application and offerings?

That’s multi-tenant cloud. That’s innovation that made us more productive shoppers. We get constant innovation in our personal lives, why shouldn’t we have it in the workplace?

The business world is changing faster than ever before. The technology and business processes that impact software are changing too. It’s imperative to have a software strategy that keeps pace with change through ongoing innovation.

Without innovation, companies become more vulnerable, less efficient and lose competitive ground. Without software innovation, your applications may not deliver the productivity gains that are needed every year to stay competitive.

Headwinds and Tailwinds to Software Innovation

Software's delivery model directly impacts its pace of innovation. It also impacts the value and disruption that comes with innovation.

The right model speeds things up. The wrong one can hold a business hostage, stifling innovation, and the software begins to set like cement.

Multiple Versions of Software

If a software vendor is managing multiple versions of software at the same time it slows down their development team. They have to divide engineering and QA resources to support older versions of the software, plus the additional rigor needed to make sure new versions play nice with older versions.

It's like building a new house while restoring 5-10 others. At the same time. With the same number of builders as if you were just building one house.

This approach divides focus between looking forwards and backwards. This innovation quagmire is one of the biggest disadvantages of on-premise software or vendors that have multiple software delivery models. If you keep spinning more plates, eventually one will drop.

In contrast, multi-tenant software has one current version of the software and all of its customers are on it. This allows multi-tenant software vendors to innovate much faster. They typically deliver two to three new releases a year with new features, enhanced functionality, and greater flexibility.

The Comfort Zone

On-premise systems take more internal IT resources to manage compared to multi-tenant cloud. This is due to the custom nature of on-premise software and the associated software-hardware-network management burden.

If many of your internal IT resources are deeply immersed in a particular technology, they tend to lose perspective or even the desire for innovation. They are simply too busy keeping the lights on. When the only tool in your backpack is a hammer everything you look at starts looking like a nail.

Cloud software shifts most to all of the management burden to the vendor, leaving your IT organization to focus on business value and strategic innovation. And that's all I have to say about that.

The Return on Innovation Investment

If a software vendor is managing multiple versions or multiple "clouds", the bang for their buck goes down. Investing in a new capability, be it an application feature or a performance improvement, requires more resources because it has to be implemented multiple times. This leaves private cloud software vendors stuck making difficult choices that should be easier to make.

By contrast, multi-tenant software investments are applied once and everyone benefits. The efficiency of multi-tenancy delivers greater ROI for customers and directly benefits the software vendor through happier customers. Win win.

Cost & Disruption

Upgrading on-premise software (or hardware) costs money and can be disruptive. This fact puts you, the customer, in a challenging predicament. Should you upgrade to get the innovation?

It depends. Is the upgrade giving you enough value to justify the cost and effort? What if you just stay on the current version? Will your end users be okay if you just continue to use the old version for a bit longer? What all do I have to do to implement and support this upgrade?

The answers to these questions can be costly, literally or figuratively. As a result many companies opt to skip a few upgrades until their users just can't take it anymore. Or until they get additional funding. But by that time you are dealing with a change management nightmare since your users basically just woke up from a five to seven year software coma.

It's like going from a flip phone to the latest iPhone or Samsung Galaxy. Learning curves aren't meant to be so steep that half the people slide back down and revert to manual workarounds.

Visibility and Insight

This aspect of innovation is simple, and quite trendy. Big Data, Data Science, Machine Learning, and all that jazz. While there remains some skepticism around those new obsessions, it is undeniable that the more you know the more informed decisions you can make.

How customers are using software is very valuable information to a software company. It's golden data that tells us vendors what is going well, where we need to innovate, and what needs attention. Adoption patterns, what do users struggle with, how long do screens take to load, how much data is there, how is one customer doing compared to their peer group.

Multi-tenant cloud software gives a vendor 20/20 vision. Private cloud makes it possible to get visibility but harder to aggregate and benchmark. On-premise software leaves the vendor blind. The smarter your vendor, the more proactive they can be and the more thought out their enhancements are.

Depreciating or Appreciating Assets

Innovation comes at a cost. The difference in software models is who pays for the innovation and what burdens it puts on the customer.

If you purchased a car 10 years ago, chances are it did not have backup cameras, blind spot indicators, or 12 airbags. If you want to upgrade to benefit from the safety innovations of the past ten years, you need to get a new car. This is akin to the on-premise software model. Buy a depreciating asset, deal with it for as long as you can tolerate, then spend money to upgrade.

In the multi-tenant cloud model, the software is an appreciating asset. New innovation becomes available with your subscription (license) and it is applied to the current and only version of the software. Typically in two to three releases a year. And features are default off until you chose to leverage them.

This model is not unique to enterprise software. Take a car, like Tesla. Today a "tune up" is parking your Tesla in your garage, connecting it to your wifi, and letting Tesla run diagnostics and upgrade the car's software systems.

Imagine that scenario in a private cloud - Tesla would have a server in the cloud for every car? And cars would be on different versions of software?

Aside from the expense, lack of aggregate visibility, support complexity and general chaos, can you imagine the liability exposure? And how would that impact Tesla's quality and brand? It's untenable.

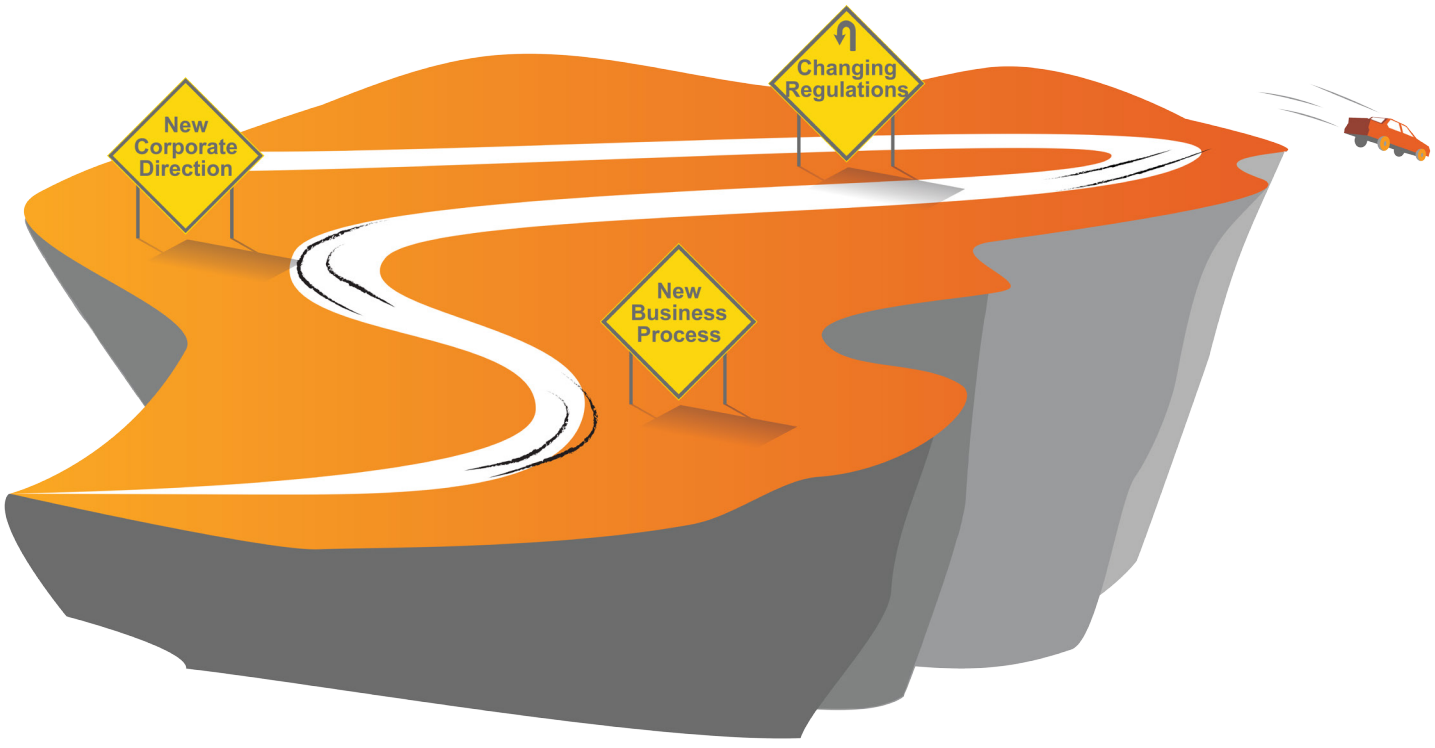
Flexibility

6

I can't help but wonder what Steve Jobs would have thought of the iPhone X (I still say "X", not 10): wireless charging, facial recognition to unlock the phone, no home button, and a silky edge-to-edge screen.

I've gotta hand it to Apple. The iPhone, as a core innovation platform, has afforded Apple the flexibility to respond to changing demands of the connected consumer. Whether it's streaming Netflix on the bus, Facetiming with the kids when I'm traveling, or using my face to pay for food at WholeFoods, the iPhone is there, keeping pace with my life.

While we have covered other important topics in this blog series, flexibility is where the rubber meets the road. And it's a very curvy and hazardous road.



In chapter one, I said that flexibility should be your north star to guide your software evaluation process. Without it, challenges mount every year that your company uses the same software. Change is not always predictable but it is certainly coming. Flexibility is how you support that change.

Many people believe flexibility is being able to customize software to your specific needs. This is not wrong. Software should be tailored to your business needs. After all, we're all "special", right? But the wrong type of software customization can lead to inflexibility.

If you hire or contract enough developers, you can customize software to meet your exact needs. These are your needs today, but what about a year from now? Will your needs stay exactly the same? What will need to change?

Even with predictive tools, we cannot control where every change will come from, or how it will impact us. So we are all constantly reacting or preparing for the changes we predict or know about. A software application should be no exception.

Four Flexibility Factors

Here are four questions you should ask to determine if the software you are evaluating is truly flexible to keep pace with change.

1. Is it flexible to meet the needs of different users or applicable business units?

This is especially important for large or global organizations. As they design processes to meet applicable standards, organizations inevitably come to reality that only some things can work the same way across different departments and/or in different countries.

Many processes need autonomy to support product classes, local business value, different business units, and even regional specific needs. The ability to take a global view along with the flexibility to tailor it locally: I call it GLOCAL ….Trademark.

Take Proposition 65 for example. Prop 65 is a series of chemical safety standards specific to the state of California. It is unique. If your software is meant to support compliance processes, you should not add complexity to core and common functionality to support outlier needs like Prop 65.

Flexibility should support a healthy balance of standards with autonomy. And if this balance managed the right way, it can have a great positive impact on usability. Why should a user in Germany have to do things or see data in the software to support processes that have no relevancy to them?

Governance can make things go faster, but too much governance can stop progress dead in it's tracks. Software that is not easily flexible leads to the iron fist of governance which can cause you to regress.

2. Is the flexibility enabled via configuration, custom code, or a combination of both?

Configuration means point and click adjustments of the software. Most people can point. Most people can click. So configuration is always the fastest and more cost effective approach. But software configuration has it's limits.

Some software companies offer an software development kit (SDK) or development framework to support needs that go beyond native configuration. This is a good thing. And as a recovering Texan, this makes me think of the saying, "It's better to have a gun and not need it than to need a gun and not have it."

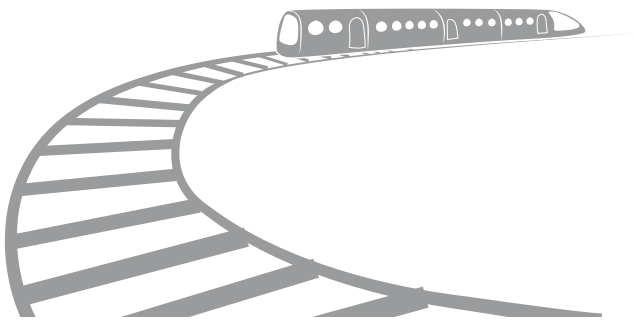
But beware, code is code. Code written for a bespoke need can become a liability. For example, you need to make some adjustments to your software that requires code modifications. But it was written by some guy that is now seemingly in a witness protection program and can't be found. You either need to do a full rewrite or make a dangerous amount of assumptions. Business disruption is almost certain.

If custom code is how your software is "flexible" than it's time to get modern software. Cloud software.

3. How easy is it to change, modify, or enhance the software?

As changes come your way, sometimes you have advanced warning. Other times it's a five-alarm fire. You need software to be agile to keep pace and avoid 3rd degree burns.

If it takes six months to make material changes to your software by then the train has already left the station and your software's inability to flex has just impacted your top and/or bottom line.



To help assess this, put the poor guy or gal demo-ing potential software on the spot by asking them to make a change to a workflow or a security setting. If it's a cloud vendor, look at the release notes for the last release to see what enhancement came out that support greater flexibility.

People tend to look at cloud software releases as just new features coming or enhance functionality. Those tend to grab the attention of customers the most. But flexibility enhancements far exceed the impact of just some new flashy feature. It's like getting better suspension in your car to stick the sharp turns.

4. Can the software application be extended?

Changes that require extending a software application can be difficult to navigate. If you can't extend the application, you need to either get a one way ticket to the forbidden dark forest of Excel or buy yet another technology to integrate into your current system.

The extendability of on-premise software is accomplished via custom code or paying for a new module. Sometimes that module needs to be integrated, as it may use different technology from the same vendor.

With cloud software, the big difference isn't about whether it's private or multi-tenant cloud. What matters is if the application is built on top of an extendable application platform.

You may be wondering - what is a application platform? How can you tell a good platform from a bad one?

Thanks for opening up another can of worms Frank (ya' jerk!).

Applications and Platforms

When evaluating software, you should know if the solution is a purposed built software application (built only for what it is supposed to do) or is it a software application built on an application platform.

An application platform is a set of capabilities and an infrastructure to build one or more applications. Either applications packaged by the vendor or custom built by customers or services partners.

To be clear, I'm not talking about infrastructure platforms like AWS, Azure, or Google Cloud. Otherwise known as Infrastructure as a Service (IaaS). That is simply where you put an application or an application platform.

Many vendors will refer to their software as a platform, so here is a quick way to get to the truth.

- How many applications have been built on the platform?
- Are those distinct applications or just modules or features of a single application?
- Can customers build their own custom applications on the platform?
- Do you have any partners building applications on your platform?
- How do you separate your applications on the platform?
- How does licensing work if I want multiple applications on your platform?

The answers will reveal more than you might imagine. A good application platform is versatile and flexible. It also enables your organization to change or modify the applications through configuration.

7 Conclusion

On - Premise Software

REALLY?!? Why should anyone read a “conclusion” chapter in a blog series that will clearly suggest that a multi-tenant cloud software strategy is the only way to go? Haven’t you already shared your conclusions Frank?

Yes and no.

Yes.

On-premise software is approaching extinction. **Look, I want to save the Northwest Borneo orangutan as much as the next person. But in the case of investing in software for your future, on-premise or custom built software should not be considered if you can avoid it.**

Don’t waste your time and money on an outdated strategy. Don’t put your company in jeopardy by investing in technology that will chip away at any competitive advantages you believe your company has.

The best option for future success is cloud software. Period. This is a debate that should have ended five years ago.

No.

Not every company is compelled to use the best tools out there. Sad but true. And it is not lost on me that moving to the cloud is easier said than done given certain company dynamics and the incredible force of inertia.

Budget cuts are in the way. Lack of education is in the way. Long tenured preferences are in the way. Change management is in the way. Hell, I'll say it, short sighted thinking is in the way.

Is your company looking over its shoulder, looking at its shoelaces, or looking at the horizon?

There are also many on-premise, siloed, custom-built applications that have seemingly no direct "costs" anymore. Maybe if the application doesn't change, the world will stop changing?

Sadly, I don't think it works like that.

And you probably have four other applications that basically do the same thing, but there's currently no way to consolidate them into a single system or process.

Sunk costs, missed opportunities, a long tail of indirect costs, and a never ending game of technology whack-a-mole. What is the real difference between avoiding a new cost versus preventing efficiency and enabling improvements?

Be responsible to your organization's future success and always yearn for a better way. If you don't, your competition will and then crush you.

Cloud Alone Will Not Solve Your Problems

I am not alone in the belief that it is not a question of IF you will move to cloud, it is WHEN. But going from "WHEN" to "NOW" needs to be looked at realistically and holistically. You need to break down the scar tissue built up by legacy on-premise software, the fear of change, and the growing pile of bespoke applications that is so large your company could be featured on the show Hoarders.

Declutter the mess! Throw away that owner's manual to the IBM PC Jr. you had 30 years ago. Software isn't a collector's item.

But how does your organization execute a successful transition to the cloud. Ideas are a dime a dozen but execution is in short supply. Just turning off a legacy application and implementing a modern cloud application will not magically fix all your built-up problems and give you the value that cloud software vendors promise.

First, you will need to take care of your legacy "baggage." Every company has processes and data. Often those processes and that data have been influenced by how you are using technology.

For example, if you are a manufacturing company, you have processes for how you work with your suppliers. With legacy software, your collaboration with suppliers is done over the phone or through email….and let's include fax machines just for good measure.

With cloud software, your collaboration with suppliers could be done in a shared cloud application you both access securely from anywhere. That's a massive difference.

But a move to the cloud needs to be accompanied by process improvements and process simplification. It also should be the event that forces you to clean up your data. You can implement the best application out there, but if you pump a ton of poor quality data in it, your application will smell as bad as the data you just put in it.

Less is More

The is no debate; companies should aim to reduce the number of technologies, applications, and software vendors they manage. If you adopt a new technology, hopefully it means you can decommission more than one legacy applications.

Yep, I'm talking about application rationalization. Less applications doing more for you…in a better way. If you are part of an IT organization, you are probably already part of your company's application rationalization to clean up the mess so you can execute on digital transformation.

For those not in IT, application rationalization is a great thing for you. It means less applications for you to learn, use, and deal with. And if done correctly, better and easier-to-use applications. The phrase "variety is the spice of life" did not have software in mind when it was coined.

Buyer Beware

Getting back to the conclusion. Hopefully by now we are all aligned on a cloud-first strategy. So let's address the challenge of finding the best path across different types of cloud(s).

This blog series has greatly simplified "cloud" to avoid turning into a novel. Private cloud vs. multi-tenant cloud. I wish it was that simple. Fact is, there are more cloud varietals appearing every year. Even a blend varietal with "hybrid-cloud," which attempts to appeal to all sides.

Just like hybrid cars or hybrid bikes, this software type isn't great and will be short-lived as an "transition" solution. Hybrids are the bridge between gas and electric engines. Or worse, they are a cross-over between a road and mountain bike - mediocre at both.

In prior chapters, I've made claims of why multi-tenancy is a better path than private cloud software. I won't belabor those points. Generally speaking multi-tenancy is better for security, innovation, scalability, predictability of cost, and makes the vendor smarter.

But the approach, experience, values, and talent of specific vendors matter too. There are awful multi-tenant software companies and there are strong single-tenant cloud software companies out there. So my advice is to read the fine print and know the details. Do your homework so you can ask the right questions.

Look at the pedigree of the vendor's leadership. Understand how decisions get made. Search for hidden costs. Get references. Look at commitment to a focused strategy (i.e. avoid vendors that have an on-premise, single-tenant, and multi-tenant cloud strategy).

And understand how they execute when it comes to rapid innovation and increased flexibility.

In the next, and final, chapter of this blog series, I will provide a series of questions to ask when evaluating software vendors. This will help you smoke out imposters and isolate vendors that can really deliver value for today, next year, and 10 years from now.

Just like a home purchase, choose software that you can grow into and will last longer. Because moving sucks!

8

20 Questions You Should Ask Vendors

RFIs, RFPs, and pilots. Bake-offs, ROI/TCO calculations, demos, and feature comparison checklists. These can all be effective software evaluation tools, but not every company uses them the same way.

When making a strategic software investment, the evaluation process often varies but the objective is the same: find the best solution for my company.

We encourage customers to use various techniques to both de-risk their technology investments, and expose the weaknesses of vendors who might appear strong on paper but can't match up on a real-world use case.

To establish to a common ground, let's start by shifting the buyer's mindset.

Approach software evaluations like you are interviewing a potential employee. Look as their past. Their pedigree. Get references, ideally blind. And get to know them to assess the impact they can make in the future, how their goals align to your company's, and culture fit (open API, fast, scalable, flexible, etc).

Don't make a bad hire, as they are more costly than doing sufficient due diligence. To help; below is a list of 20 general questions for software vendors. They are market and application agnostic.

20 QUESTIONS YOU SHOULD ASK VENDORS:

1. How many versions of your software are you supporting?
Best answer is one, as this will give you the fastest pace of innovation.

2. What countries do you have users in?
The more the better as this speaks to scale and global performance.

3. What is your upgrade or release process? How often do you do it?
Flush out any hidden costs or potential disruption as well as pace of innovation.

4. What are the ongoing costs in addition to licenses?
Lower costs are better but also assess cost predictability.

5. How many customers do you have, and what is their collective volume of data/ content?
More the better to demonstrate scale and execution.

6. What is the background of your leadership team?
Leadership experience in software matters. The more patterns they've seen, the better.

7. How many different software delivery approaches do you have?
Best answer is one. Red flag if a vendor has on-

8. Where is the development team based?
Communication is integral to success software companies. Needs come from customers go to product designers, which go to architects, which then get coded by developers, which are then tested by QA teams. This relay can lose fidelity if communication is not crystal clear.

9. Is your company profitable or cash flow positive?
A software vendor in good financial standing has the means to invest more than those that are not.

10. What is the ownership profile of your company?
Are they public, VC-backed, or private equity owned? Public is safest as nothing is hidden. Private equity is riskiest because a banker is captaining a software investment ship.

11. How quickly is your revenue growing year over year?
While customer and employee growth is also good to understand, the numbers can be misleading as other variables can be at play. Revenue growth is the purest motivator for ongoing investment for a software company.

12. Does the software offer an open, public API at no additional cost?
Systems are meant to talk to each other. In 2017, an open API is a must. If a vendor charges for the API then what else are they going to nickle and dime you on?

13. How is the software modified/ customized? Is it by configuration, code, or both? If both, find out what the configuration is used for and what the code is used for.

14. Is the application built on an application platform or is it a point solution?
Many vendors will call an application a platform. The quick test is to ask how many applications are on the same platform...and don't let a vendor call a module or add-on an application.

15. What customer use metrics does the vendor track?
Performance should definitely be one.

16. Is the licensing perpetual, named, or concurrent?
Is it based on data volume, logins per time period, or something else? Perpetual is usually for on-premise. Some vendors offer concurrent licenses, and watch out for those. If you exceed your concurrent licenses due to an event, you may be looking at large "roaming" or overage charges…..or worse, your users can't get in.

17. Where is the software hosted?
The best are AWS, Microsoft Azure, Google Cloud, NTT data centers, etc.

18. Does the vendor provide transparency into the status of the service?
How will you know if the system up, down or sideways? Pure cloud vendors will often provide a public site that details the status of the service and will report on any disruptions to the service. Here is an example: trust.veeva.com

19. Does the vendor have a robust partner ecosystem?
System integrators and other 3rd parties can play a significant role in the tuning of a SaaS solution. Don't be trapped and sole-sourced by your vendor.

19. cont... Make sure that there is an ecosystem of partners who can help you to configure and fold the service into your business environment.

20. What is the vendor's customer renewal rate?
In the world of SaaS, customer renewal rates are a proxy to customer satisfaction. If a customer is unhappy with a service, then they won't renew their subscription with a vendor. Also, if a vendor is lucky enough to have "negative churn," then they are expanding their footprint within accounts, as opposed to shrinking or not renewing. Negative churn is the best indicator for both customer satisfaction and adoption.

Armed with these 20 questions, any software buyer can select the best software vendor for their company's needs.

industries.veeva.com

About the Author

Frank Defesche began his software career at Trilogy Software in Austin, TX, an on-premise software company. In the summer of 2000 he joined salesforce.com as one of their first consultants. In a world dominated by on-premise and home grown software, he was faced with the challenge of translating traditional software processes to an emerging cloud paradigm. He was part of the cloud's first chapter and has lived it ever since. He currently serves as the SVP and General Manager of Veeva Systems and is responsible for expanding Veeva's solutions to the consumer goods and chemical industries.