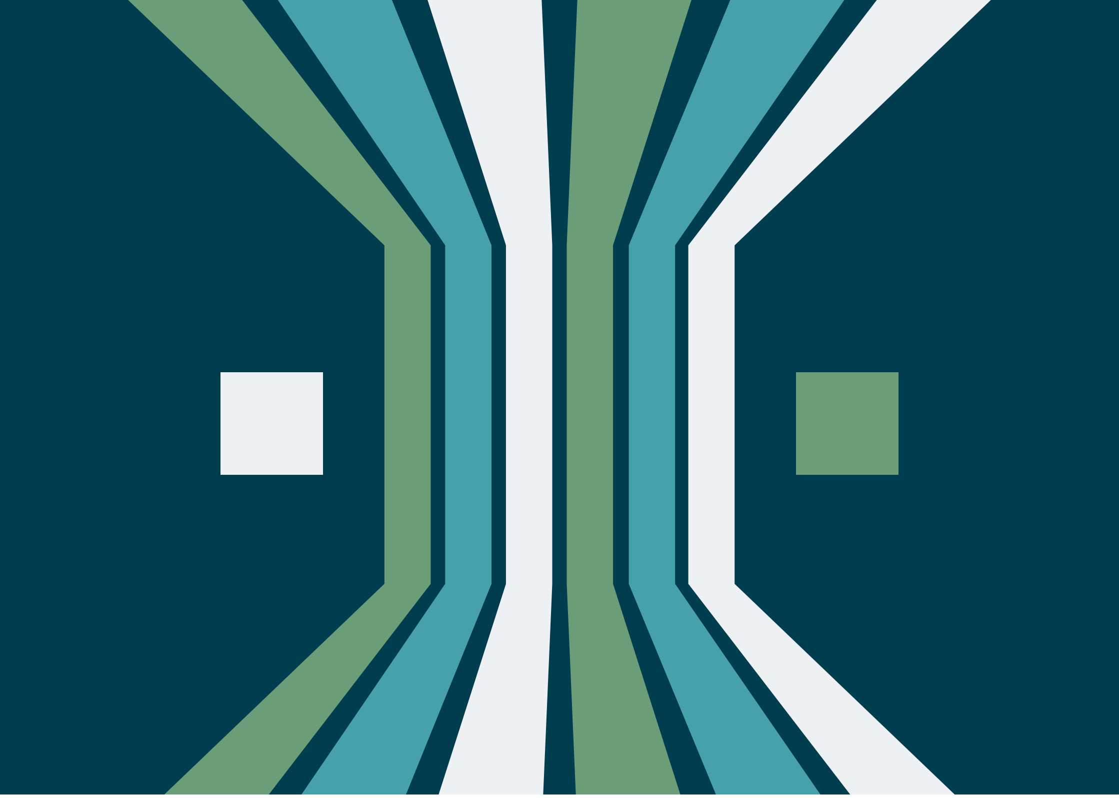


Build vs. buy

A strategic framework for evaluating third-party solutions

/thoughtworks

Introduction

When your organization needs new software capabilities, the process begins with a deceptively simple question: should we build the new capabilities in-house, or should we buy a ready-made solution?

When you buy third-party software, you gain proven capabilities quickly, at the cost of customization and control. When you build, you get the opposite: exactly what you want, but at greater expense and effort.

When your need is clear, immediate and has a relatively low potential impact on the organization, it makes sense to answer this question quickly, and get started along whichever path you choose. But for larger impact areas or capabilities that are likely to have a far-reaching and long-term impact on the organization, build vs. buy becomes a far more nuanced decision. It demands significant evaluation and carries serious risks if you don't make it with the right level of care and diligence.

When you make a build vs. buy decision, you are really asking the question, 'is what I need available as a pre-built solution, or do I need to build these capabilities myself?'. That opens up a number of further questions, such as:

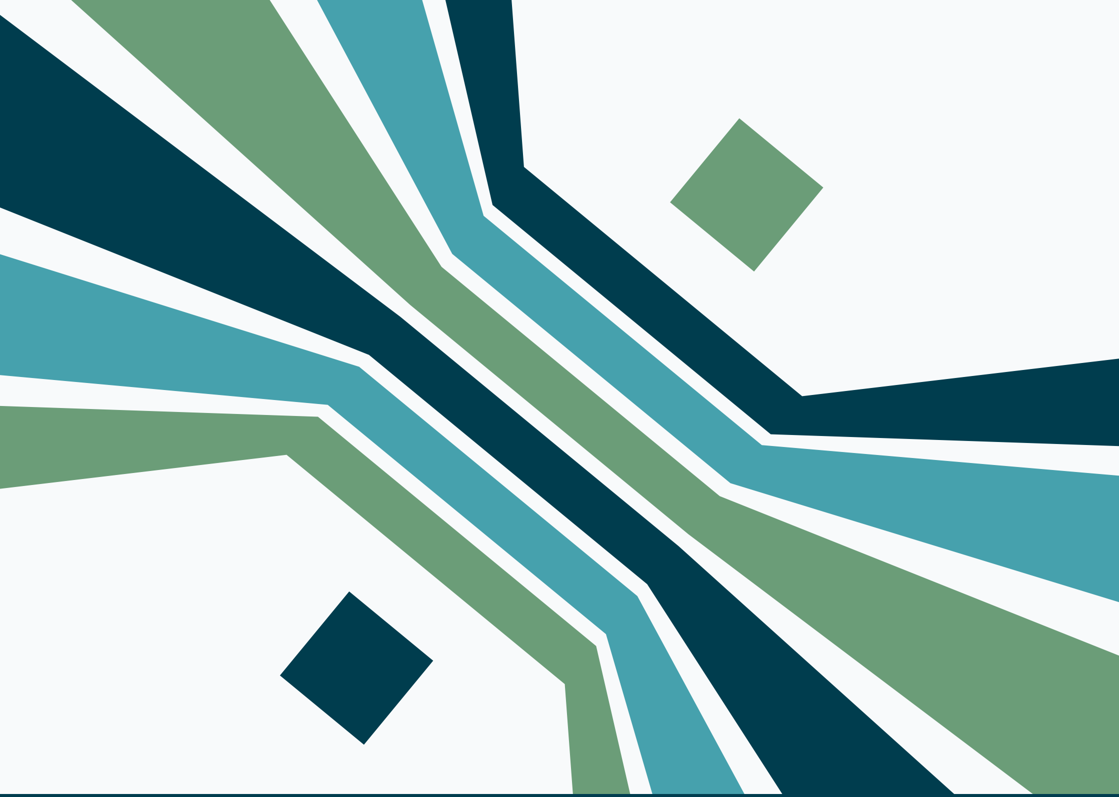
- Is the capability we're looking for a key differentiator for our business? Do we risk yielding competitive advantage or control by using a third party to acquire it? How will we need that capability to evolve over the next three to five years?

- Can a third-party solution meet our specific technical needs? For example, does it offer the level of security our organization, its customers and local regulators demand?

- Which vendor(s) can realistically deliver the level of capability and service we need? For how long are they likely to support the solution?

That’s the dilemma with build vs. buy. It looks like a binary decision, but in reality it’s a multifaceted decision with radically different outcomes depending on your needs and current situation.

In this paper, we’ll break down the third-party evaluation process, examining what it really takes to select the right partner and vendors. We’ll also take an in-depth look at the questions you need to ask to determine whether building or buying is right for your business.



Defining your needs

Defining your needs

Before you even consider evaluating third-party solutions, you need to ask a few important questions to determine if going with a third-party provider is the right choice for the capabilities you want to gain.

Generally, capabilities can be organized into one of two groups:

- **Commodity capabilities** are required by your company but aren’t unique to you. Common examples include payment and payroll tools. These capabilities offer little to no competitive advantage, so it makes sense to follow established best practice and adopt the industry standard

- **Differentiator capabilities** are how your company differentiates itself in the market. In competitive markets, higher investments into these capabilities are justified, as they directly help a company stand out and operate in unique ways

When considering what is a differentiator or a commodity, it is important to have a clear idea of your own business context and what this means.

A commodity capability will be provided under the assumption that you will adapt your processes to that particular vendor’s definition of industry ‘best practice’. From a strategic perspective this can alter your ability to deliver value to your customers. Often commodity capabilities are chosen when following the same process as the rest of your industry does no harm.

There can be other aspects beyond the traditional value drivers which make the capability a differentiator in your context. This can include your operating model and your culture. These areas can be harder to identify as they need to take into consideration the way you organize and get work done and what behaviors you wish to encourage or dampen. Commodity capabilities will force you to follow their version of these best practices too.

Another consideration is around granularity. A common mistake is to scope at the level of the package resulting in capabilities which are too broadly defined. For example, a Customer Relationship Management system or an HR system needs to be mapped at a more granular level, closer to the value chain. When organizations do this, they often discover a mix of smaller commodity and differentiator capabilities within the larger solution itself.

For example, at Thoughtworks, our strategic capability is based (in part) on our consultants’ ability to flex their schedule to meet the client’s needs. If the people team’s recruiting system requires interviewers to register a consistent block of availability each week, that impacts the strategic value in both directions because it puts client needs and the recruiting scheduling process needs in direct opposition. Quickly hiring quality candidates is also a strategic capability; realizing the impact on our strategic value ultimately drove additional investment in the HR recruiting tool.

Understanding the full impact of a buy decision is key to avoiding unintended impact upon the key strategic differentiators which drive your business. While it is typically obvious how the vendor’s best practices will affect your ability to differentiate your strategic capability within the market, it is not always so clear how your operating model and culture may be subtly affected. For normal cost-of-business processes it may not matter at all, but when your operating model and culture directly support your differentiated capabilities, it becomes critical.

Wardley Mapping
A good exercise to help you understand whether the capabilities you need can be categorized as commodities or differentiators is **Wardley Mapping**.

Wardley introduces the idea of evolution, the movement from novel to commonplace. Capabilities begin life as novel differentiators. They're new, there are few third-party options available and best practice around them is still in the process of being defined.

Over time, as that best practice becomes clearer, demand rises and third-party solutions mature, they become commodities. Those are the capabilities where it makes the most sense to go with a third-party provider, and the ones that are worth spending the time evaluating in detail.

Wardley Mapping encourages mapping at the value chain level. Armed with this information, it is possible to draw a clear boundary around which specific capabilities the vendor should and should not provide. This enables the **Bounded Buy** strategy which prevents the **Vendor King anti-pattern**.

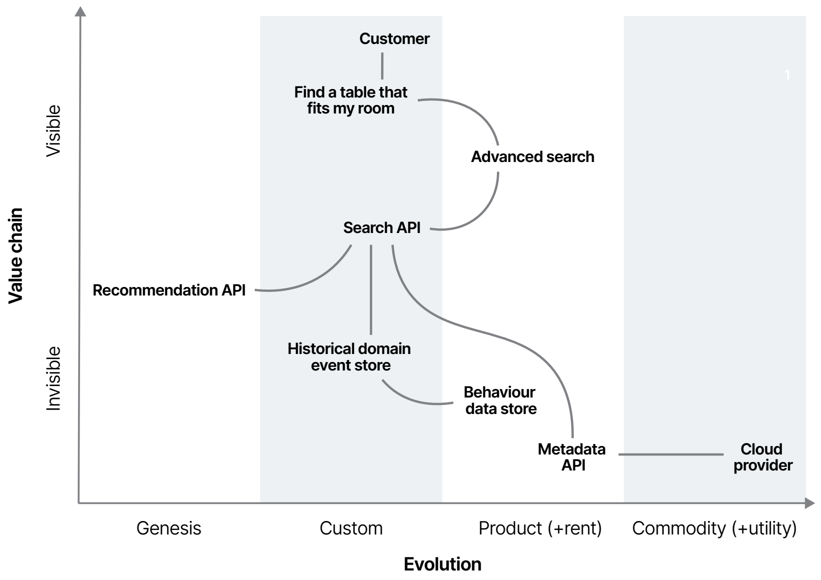


Figure 1. Example of Wardley Mapping

Replacing or extending existing solutions

On a regular basis we should re-evaluate our landscape to see if there is a third-party solution that makes sense. The industry moves quickly. An evaluation you did a year ago might have changed with new features and different competitors. Your company may have also changed or business priorities shifted, and that unique functionality you thought you had to build didn't become a reality. It also could be that pricing has changed, making it more appealing to use a third party.

There is always a tendency to want to throw out the older legacy solution in favor of something new and shiny. That can be a valid choice, but before you can make a pragmatic decision, you should also consider whether you could extend the life of what you already have in place by:

- **Bringing your solution up to a modern standard,** so that it can work in your DevOps workflow. This might involve putting it in a container, automating deployments or figuring out how to horizontally scale

- **Customizing or extending the solution** if there is a way of adding behavior that might negate the need for a new solution. For example, if there is an API from which we extend new functionality, or if the solution has customization abilities. This has to be traded off with the maintenance expense of the new code

If you've determined that a third party can provide the capabilities you're looking for, that acquiring them as a commodity makes sense for your organization and that extending the life or functionality of existing solutions isn't the right choice for you, then you're ready to plan your third-party evaluation.

Planning a balanced third-party evaluation

Evaluating third-party solutions is an extremely challenging task. There's far more to it than simply comparing offerings and weighing out benefits. In practice, a thorough evaluation requires you to do three difficult things at once:

- **Predict the future:** You're not just looking at how well a solution can meet your needs today. You must also consider how well it will meet your evolving demands over the next three to five years, and factor in the provider's own roadmap for the solution

- **Balance the needs of a huge number of people:** Chances are, a third-party solution will encroach on (and ultimately replace) a few products that people across your team currently like using. Considering your team's needs as a whole against the experiences of those that won't welcome the change is a tricky balancing act

- **Forecast long-term costs and ROI:** Acquiring new third-party solutions typically involves getting tied into long-term contracts. Financial evaluation needs to consider how the value delivered by a solution may shift over time and how that may impact your budgets and bottom line

It's a complex and time-consuming process, but it's well worth investing the time and resources to get right.

Evaluating with a cross-functional approach

Evaluating with a cross-functional approach

Once you've identified buying a third-party solution as the right path forward, it's time to assemble your evaluation team. Building a representative cross-functional team is essential for ensuring that the solution you choose meets the critical needs of many different stakeholder groups.

Which stakeholders are involved in the evaluation will depend on the capability being evaluated. But cross-functional teams are typically comprised of stakeholders from the following groups:

- **Finance:** Ensures the cost of the tool aligns with budgets
- **Product managers:** Give detailed input on the specific requirements for the tool
- **End users (or proxy for end users):** Explain how they want to engage with the tool
- **Engineers:** Offer perspectives from those who will have to integrate and customize the software
- **Infrastructure and IT operations:** Responsible for deploying and supporting the tool
- **Back-office operations:** The team who will have to administer the tool
- **Compliance and security groups:** Ensure the tool doesn't expose the organization to increased risk

Mapping features against functional needs
Feature analysis is one of the biggest traps in software evaluation. Organizations get caught up comparing huge lists of features, spending hours looking at capabilities they don't even need and ultimately selecting tools that offer the most features, instead of the ones that best meet their core needs.

Instead, your cross-functional team should work together to define a short, clear list of must-haves for your new solution. That list will shape feature analysis and keep it centered on the most important capabilities.

At this stage, your team must also consider the roadmaps of the products they're evaluating. What will you need your chosen solution to be able to do in the future? Which vendors have the best plan to effectively enable and support those things? Of course, the future won't always be clear, but good vendors should be able to give you a clear view of their roadmap for the features and capabilities that matter most to you.

Make developer experience a key area of focus
Another common evaluation mistake is failing to realize the importance of developer experience when selecting a third-party solution. Your developers will ultimately be responsible for deploying, integrating and maintaining your chosen solution. So if it doesn't meet their needs, it's irrelevant how well it meets the needs of end users and lines of business.

Here is a list of important things to look out for that can make a solution easier, more intuitive and ultimately, more engaging for development teams to work with:

Web API	Don't consider software without it. Ideally, the product should be built in an API-first manner.
Push / streaming API	The ability to subscribe to changes in data is particularly important for caching and for integrations between systems, to avoid performance and availability risk.
Data accessibility	All stored data should be available via automated methods to simplify management and integration.
Batch API	For systems dealing with large data sets, e.g. analytics, you may want a way to periodically get a snapshot of the data.
Technical documentation	Up-to-date documentation (ideally with code examples).
Querying	Good options for filtering, sorting and paging data.
Versioning	Releases managed by a sensible versioning scheme. How long is the software going to be backwards-compatible?
Backwards compatibility	Can the solution easily work alongside your existing software and platforms without negatively impacting maintenance and upgrade costs?
Command line interface	To control the application, particularly for provisioning and adjusting resources.
Client library	Multiple languages, for easy developer use.

Experiment support	Does the software lend itself to an experiment-driven approach? Are we able to show multiple different experiences to different users to support development and test processes?
Integration	Can the software easily be incorporated into the current environment and rolled out incrementally in an agile manner?
Upgrade strategy	How often are releases made? How are they delivered?
Environment options	Does the third party provide QA and staging environments, and a method to test in production with a test account?
Multitenancy	If the system is going to be used by multiple teams, is there a way to segment data so they can work autonomously?
Open standards	Being based on an open standard is a good sign of the technical maturity of the company. Relying on Open Source standards may also increase the willingness of engineers to work with and contribute to the product.
Community	Is there a large community of users, and can we easily search answers to questions? Can you hire engineers with skills in the solution?
Optionality	Can the product or service be easily used to create new capabilities in the future that address new business requirements as they emerge?

Covering your technical needs

Covering your technical needs

When you're evaluating a third-party solution, every function has their own needs to consider. That's why you created your cross-functional team.

Typically, technical needs such as security, infrastructure requirements and operational IT demands tend to fall through the cracks because functional needs are evaluated first. To avoid serious challenges once implementation begins, it's important to map out these technical requirements as part of your evaluation and ensure that the solution you choose ultimately meets as many of them as possible.

Listed below are some of the most common ones. Ideally, you'll want to assign metrics and service levels to each one as part of your evaluation and selection process.

Availability	What level of availability does the solution need to maintain? How can you verify that before selection?
Performance	What does the performance profile look like during normal operation? How does load from other customers affect your throughput? How well can the solution scale and adapt to increased load?
Accessibility	Does the solution's UI meet your accessibility requirements?
Compliance	Does the solution meet all relevant security and data protection requirements?

User experience	Run user testing in the evaluation and get feedback. How hard is it to extend and customize the user interface?
Disaster recovery	How does the system recover from a disaster? How is data preserved?
Support availability	If something goes wrong, who can help? How quickly can they get you up and running again?

Evaluating security and compliance
Security can be one of the biggest reasons to buy third-party solutions. As general resistance to the cloud has fallen and IT teams have accepted that on-premises doesn't mean 'more secure', many teams have come to love off-the-shelf solutions for the security advantages they deliver.

First, when you buy a ready-made solution, you get proven tools with an established security footprint. Most of the work to secure these tools is done and the efficacy of their security is well known and documented. Second, when you work with a third party, you have additional bodies on your side if things go wrong. If a SaaS solution goes down, the vendor's team can jump in to bring it back up and remedy any potential security issues.

From a risk management perspective, using third-party vendors can be a form of risk offload. In the case of open source software and standards, you get the benefit of the community's investments in threat modeling and vulnerability remediation. Leveraging these can help you demonstrate due care and due diligence to auditors and other interested parties.

But that doesn’t necessarily mean all third-party solutions will meet your security needs right off the shelf. As part of your evaluation, it’s important to follow these steps:

- **Analyze the sensitivity level of the application you need** and its data. By following a concept like NIST’s Impact Level model, you can determine the right security control baseline that you’ll need to assure and uphold. That will help inform your infrastructure decisions too. For example, if your application’s impact level is classified as high, you’ll need to use an IaaS provider’s public sector certified environment, like AWS GovCloud

- **Ask for security content and accreditations.** Start by asking the vendor for their security whitepaper. This should include discussion of their threat model and other security considerations. For SaaS providers, you will want to review relevant attestations such as PCI AoC/RoC, or SOC 2 Type 2 reports

- **Make sure your contracts meet regulatory demands** by building in relevant pass-through language as required by regulators

- **Be aware of potential security showstoppers.** The most common examples are when a vendor does not support appropriate standards-based single sign-on, federation and security in transit technologies

Evaluating infrastructure requirements
The range of infrastructure concerns varies significantly depending on the build vs. buy decision, as well as the type of product considered for purchase. At one end of the spectrum, SaaS products typically encapsulate all infrastructure concerns within the product offering. At the other end, should you choose to build you will be responsible for the infrastructure for supporting the capabilities.

There are two key points to keep in mind as the build vs. buy decision is contemplated. The first is to evaluate the various solutions (whether SaaS, a product deployed on-premise or a custom-developed solution) over a consistent set of criteria including (but not limited to) availability, resiliency, recoverability, service-level agreements (SLAs) and cost.

The second is that the actual usage patterns of the capability in question will have non-linear effects on the outcome of these evaluations. While initially the decision is rightly made based on estimates of usage, understanding the breakpoints which tip the balance toward one solution or another will guide when the decision should be revisited at an appropriate time in the future.

Operating the service
It is also important to consider the ongoing management of an application, and understand what the administration experience will be like. Unfortunately, we see a lot of applications that focus on richness of user-facing features, only to fall down in this area, putting additional pressure on already overburdened IT functions. This is also something that can be underserved in the prioritization queues for custom applications, and it is worth ensuring stakeholder groups include operations team members in both cases.

Beyond the compliance and security concerns mentioned above, this includes things like:

User account management	How are provisioning, deprovisioning and archiving handled? Can this be readily automated? Are there circumstances when a user external to the organization may need access and how is that accommodated?
Teams and groups	How easy is it to set up and maintain structures for teams and groups within the tools? Is this flexible enough to represent the various ways your business units are organized? To what extent can management be delegated or shared?
Settings management	Is there a flexible model for the various controls that are provided, allowing settings to be enabled or disabled at various levels according to the team structure? Does this cascade, and does it also allow for cross-organization sets? Do admins have the ability to find and fix all end-user settings? What data can they see and what protections are there to help avoid unintentional overexposure?
Audit	Clear auditability also has a role in support of end users, helping admins confirm which actions led to which results.

Reporting	Be cautious of applications that share vanity metrics, highlighting only the most active users or the most successful content. Reporting needs to be in the service of making adjustments to improve the application, and robust reports should include all aspects of usage, highlighting both immediate issues and errors as well as longer-term trends that require action.

When it comes to vendor support for building on top of an application, there is an intersection of operations and developer experience. Key things to look out for include the approach to providing sandbox or sub-accounts for development and testing, and the model for API authentication. Using an OAuth model, where an extension acts on behalf of a user, often makes it easier to attribute actions during an audit. Those that require administrator-level 'god account' access can make it harder to see who did what, as well as introduce a weak point from a security perspective. Products that rely on third-party application integration systems to provide connectors can readily advertise integrations across a wide selection of other applications, but this can also result in even more complexity from an operations perspective. Unless you already use and manage the same platform within your ecosystem, we recommend approaching these kinds of integrations with caution.

Vendor assessment

Vendor assessment

When you choose a third-party solution, you're not just selecting new software, you're picking a partner that you'll have to work with for years to come. It's just as important to evaluate vendors as it is to evaluate the software and capabilities they can offer.

During that assessment, it's important to look beyond the status quo. It's not just about who can offer you the most predictable, proven solution and consistent service. It's also about identifying who's innovating, whose capabilities are actively improving, and who has the strongest roadmap — both for their solutions, and as a company.

Most assessments are built around a combination of these nine criteria:

Trajectory	Ideally, you want to pick a vendor that you can form a long relationship with, even over the lifetime of a relatively short contract. You want to pick a company that's actively investing in their product, innovating and responding to changing market demands.
Influence	Seek a partner that will listen to your feedback, where you can influence the future of their solutions and feed into their development roadmap. This is an area where it can be advantageous to work with smaller, growing providers.

Financial stability	You need to be sure that your partner will continue supporting the solutions you buy. Checking their financial position and stability can help you ensure they'll still be around when you need them.
Size	Being small and nimble may be an asset in some situations, if you're looking at an experimental technology that you want to try out. In other situations, a larger, more stable partner may be preferable.
Technology culture alignment	It's a good idea to choose a partner that views and approaches technology the same way your business does. Do they share the same technical values? What are their QA processes? Do they practice continuous improvement and continuous delivery? Are your engineers excited by the company?
Culture	Does the partner share your core cultural values? Do they value sustainability, diversity, humility and any other quality that's important to you?
Transparency	How open are they about their technology and their weaknesses? Will they let you see their internal systems?
Access	Will they provide a technology partner or architect to help you access the services you need?
Quality of support	What is their support model, and what SLAs do they commit to upholding?

A common mistake companies make when conducting vendor assessments is relying too heavily on industry surveys or analyst reports. While useful, they often aren't published frequently enough to give you a complete view of the market — missing emerging innovators and not reflecting the latest developments in the space.

In other words, when conducting your own assessment, it's important to combine expert viewpoints with your own.

Proving concepts and value

Proving concepts and value

Like anything else, when you're evaluating software, there's no substitute for getting hands-on experience with the product and trying it out for yourself. A proof of concept (POC) gives you the chance to deploy software in your environment, try out its features, test its APIs and gather opinions about it from end users.

As part of the POC process, there are a few key things to look out for:

- **Development philosophy:** Engineers from both sides should engage with each other to get a feel for the vendor's philosophies. How responsive are they to developer experience and functional improvements?

- **Functional and non-functional assessment:** Pick some real current and future integration use cases, plus some of the non-functional cases. Then, turn some of the criteria into a list of experiments to run against the system. This will tell you more about the technical capabilities of the solution than reading the documentation

- **Development Total Cost of Ownership (TCO):** A lot of tools are not built for a DevOps culture and your development team will have to do a lot of work to integrate them into their software life cycle. That doesn't mean you should write the solution off, but you should consider the investment required

Forecasting ROI and TCO
The final step in a comprehensive evaluation process is calculating projected costs to buy and implement the solution, and how quickly you expect your investment to return value.

Calculating both ROI and TCO can be complex, imprecise processes, often based on incomplete information and best guesses. Here are a few tips that can help ensure your analysis generates the most reliable outputs:

- **Own the process:** Vendors are usually happy to help you define the total cost of ownership for any system replacement. They often provide tools and templates to help you calculate the ROI, especially for commoditized products. However, the true costs are often obfuscated by the vendor's package pricing practices. Customers need to own the model, all the key assumptions within the model and the decision-making process

- **Consider how strategy may shift:** The ROI model is predicated on the need for the specific capability for a period of time in alignment with the business model and strategy. It's worth considering how that need may change, in line with emerging tech and customer megatrends. A safe approach is to plan for strategic obsolescence and reduce the expected lifetime

- **Get a complete view of acquisition costs:** License fees are usually the attention-grabbing headline, but they don't represent the full acquisition costs. The ROI model should include any additional acquisition costs such as new software, hardware or expertise needed to implement the solution

- **Factor in ongoing support and maintenance:** If significant changes are required to maintain the system over its lifetime, support costs can quickly add up, so it's important to factor them into TCO calculations. Vendors and their implementation partners often provide extended support agreements, but you may decide to invest in your own capabilities to maintain the system and implement changes. Whichever route you choose, those maintenance costs need to be part of your ROI and TCO analysis

A note on implementation
At this stage, with your full evaluation completed, you'll naturally start thinking about implementation. A full exploration of how to implement third-party solutions is out of the scope of this article. But it's worth thinking about how you can balance the risk associated with implementing third-party solutions, and keep your options open if you end up wanting to change your capabilities in the future.

Firstly, it's important to consider how you're going to couple and decouple the technology. Your development team will likely have experience with **modernizing legacy technology** and understand the challenges with removing embedded solutions.

This is why there are still so many businesses powered by legacy monoliths. You can avoid that by only loosely coupling your third-party solution. Limit the number of dependencies on the vendor product and put it behind an API to limit direct exposure.

Second, take flexible pricing options where they're available. Pricing flexibility allows you to experiment with a solution without a big financial commitment, reducing costly vendor lock-in. But be aware that flexible pricing can lead to increased spend as your use of the solution grows.

Finally, check whether your chosen solution supports open networks, and can easily integrate with other solutions as your needs evolve. Many of the most successful third-party software solutions are built on an open architecture that allows plug-in components, enabling you to customize the solution and take advantage of expertise within the user community.

Conclusion

Conclusion

Selecting the right third-party solutions for your business is rarely a simple task. The capabilities you choose and deploy will impact and shape the future of your organization, your IT estate and the way your people work, so your evaluation process can't be rushed.

The process itself will always vary depending on the capabilities you're looking for and the gaps you're trying to fill. But broadly, a thorough evaluation process should incorporate all of the major steps covered in this paper:

1. **Carefully map out your needs** and establish whether a third-party solution is likely to be able to meet them or whether you need differentiated capabilities

2. **Work with diverse stakeholders to understand what they need** and build a cross-functional team to help support your evaluation process, providing as many perspectives on potential solutions as possible

3. **Define your technical must-haves** and ensure that your organization's overall technical, security and compliance needs are factored into the evaluation process — helping you spot clear deal-breakers

4. **Conduct deep vendor assessments** and look at their general roadmap, as well as the roadmap for the solution(s) you're evaluating, to determine whether they're the right long-term partner for your team

5. **Carefully forecast TCO and ROI** to help you gain a complete picture of the costs associated with your shortlisted solutions, and determine how quickly they might be able to return value for your organization

By following those steps, you can ensure that the solutions you choose are right for your business, your developers and the customers you serve, and focus your development resources where they're needed most.

The final thing to keep in mind is that capability evaluation isn't a one-off, closed process. It shouldn't end when you've selected and implemented a third-party solution. Today, frequent re-evaluation of capabilities and services helps leading organizations ensure that everything they've deployed is truly fit for purpose.

By looking back at everything you've deployed and reassessing its value and viability, you can ensure that all the needs you've identified are still being met by your chosen solution months and years down the line. If you've chosen well and built flexibility into your implementation model, you'll be strongly positioned to make changes wherever and whenever they're needed.

About the authors

Tim Cochran
Technical Director
North America
Connect on LinkedIn

Prashant Gandhi
Principal Consultant
United Kingdom
Connect on LinkedIn

Carl Nygard
Technical Principal
North America
Connect on LinkedIn

Contributions

Andy Yates
Head of Strategy
TechOps
United Kingdom
Connect on Linkedin

Peter Gillard-Moss
Head of Technology
TechOps
United Kingdom
Connect on Linkedin

Get in touch with us

thoughtworks.com/contact-us

Thoughtworks Ltd.
First Floor, 76-78 Wardour Street
London, W1F 0UR, UK
+44 (0)20 3437 0990
contact-uk@thoughtworks.com

Thoughtworks, Inc.
200 E Randolph St
25th Floor
Chicago, IL 60601-6501
+1 312 373 1000
contact-us@thoughtworks.com

/thoughtworks