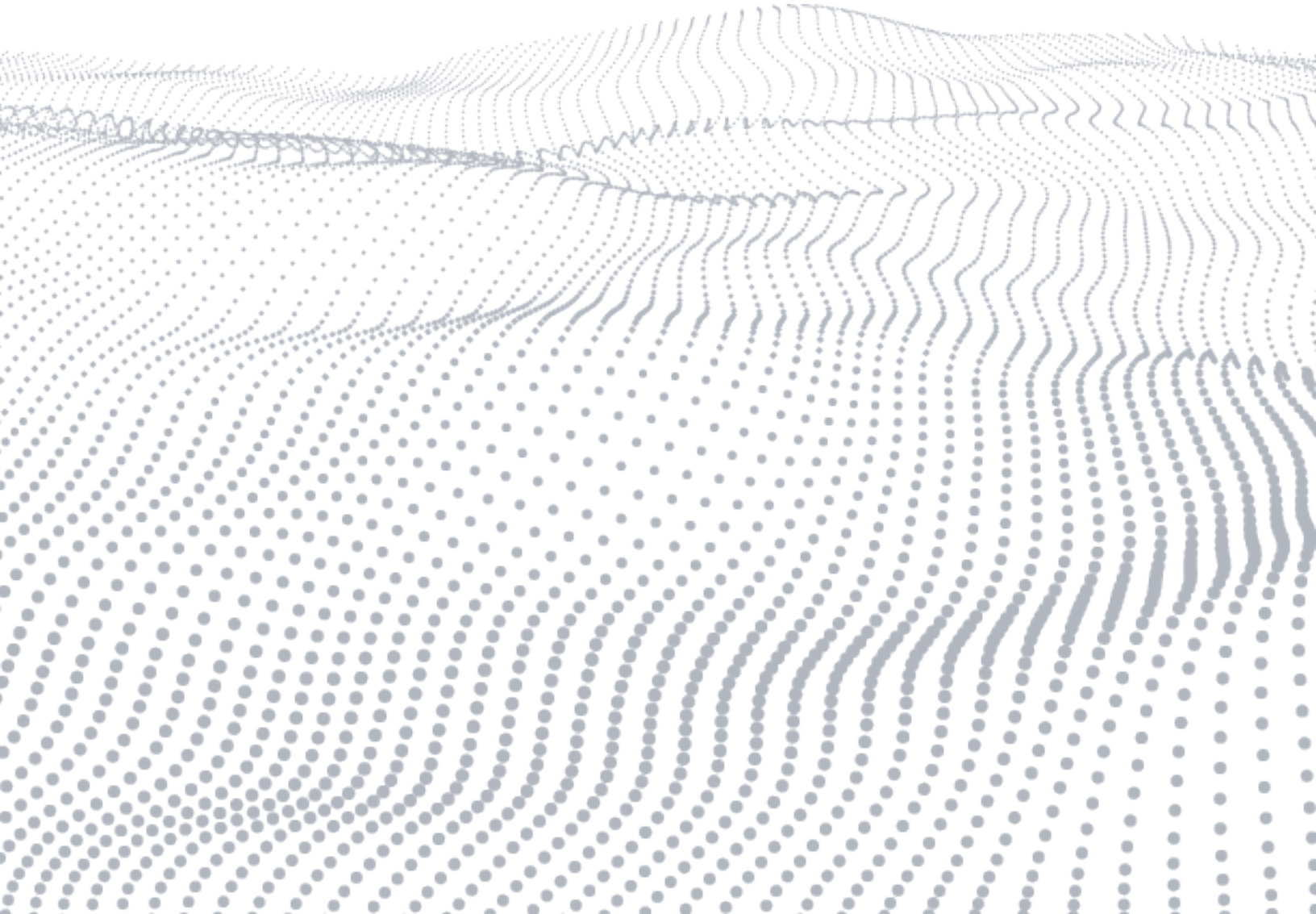
BLACKDUCK®

WHITE PAPER

Mitigating Risk in Mergers and Acquisitions

A guide to software due diligence for nondevelopers



The fundamental risk in merger and acquisition (M&A) transactions is surprises arising after the close, which, depending on nature and severity, can extend integration timelines, blow the business case—and maybe worse. This can happen in all areas of the target company, not just software, and it is one of the reasons to perform due diligence. The point of the due diligence process is to confirm the investor's premise for the deal, provide necessary information to plan what happens once the acquisition is complete, and identify issues that did not arise in earlier discussions.

The analogy of a home inspection is apt. In addition to allowing the buyer to measure for drapes, the inspection is an opportunity to discover cracks in the foundation or leaks in the roof that might change their perspective on the deal and perhaps justify some accommodations from the seller. The information gleaned allows the homebuyer to prioritize work that needs to be done before moving in. Similarly, software due diligence is critical to informing the deal as well as integration planning.

Technical debt

Understanding software development, technical debt, and risk helps nondevelopers better plan and evaluate the results of software due diligence. The white paper "Understanding Software Development, Technical Debt, and Risk" is recommended reading. In short, it states that although there is some art involved, building viable software requires good development processes. Software developed through a comprehensive process that is governed and followed will likely be of higher quality along a number of dimensions including security, third-party content, and quality of design and code. It's not a given that a less mature organization will have a well-designed and documented process. But even sound processes will only produce good results if they are followed.

Time or budget pressure can push developers or their teams to take shortcuts, such as rushing through final QA testing to meet a deadline. Generally, such shortcuts will produce a positive short-term effect—Yay, we made the deadline—but create problems that require cleanup downstream—Bummer, we need to do an unplanned patch release to assuage some angry customers. Some level of shortcutting is the norm, but it needs to be managed. The accumulated backlog of cleanup work resulting from shortcuts is known as technical debt. Unknown and unaddressed debt translates to risk.

Types of technical debt align with types of software risk.

- Failing to design **security** into applications exposes them to breach.
- Using **third-party code** with problematic licensing or known vulnerabilities exposes companies to lawsuits or breach.
- **Poor-quality** code or architectural design exposes companies to unhappy customers or poor maintainability and scalability.

Software due diligence guidance

Mitigating risk in M&A requires taking full advantage of the due diligence process to learn about the state of the target's software and the processes behind it. Advance planning is critical. The due diligence window closes rapidly, and there are many competing demands on the target's senior personnel. Efficiency is paramount. Coordinating across the due diligence team is important to keep from overwhelming the target with competing demands. A comprehensive checklist is a great starting point.

But no matter how well-planned and efficient the process is, there is always more to be learned than there is time for the acquirer to learn it. The checklist has to be focused on the scenario and investment thesis; for example, if the vision is to invest in a global sales team to dramatically scale the business, the acquirer will want to ensure that the SaaS platform and operations can handle the customer ramp. Or if the plan is to integrate the target's application with the acquirer's on-premises solution, ensuring that the open source is properly licensed for distribution is key.

The diligence team needs to be nimble, able to prioritize and deprioritize along the way. For example, if the target is found to have been lax with application security, or worse, has been breached, that puts a high priority on a deep dive into vulnerabilities in the public-facing applications. If it emerges that several products are winding down, the focus should be on the remaining products.

Much due diligence activity is based on conversations between acquirer/target counterparts and a review of documents. This is key to understanding the development organization and the processes it employs. This evaluation provides necessary information for integration planning and a look into the effectiveness and efficiency of future software development. But equally important is evaluating what's actually in the software. That might correlate with the process evaluation—but it might not. A solid QA process should produce quality software, but it may just look good on paper or only have been recently implemented, so the software may be buggy. The technical debt in the code is a review of past sins, but it is still critical to understand for future planning. Whatever issues are found need to be accounted for in future development plans or the roadmap is at risk.

Evaluating processes and the organization

Interviewing team members and evaluating documentation is the only practical way to assess a development organization and its processes within a tight window. Strategic acquirers are in a good position to execute this facet of due diligence themselves. For starters, they have the technical personnel on staff. They come into the process with an intimate understanding of their own culture, organization, and products into which the target will merge, so they are in the best position to judge the fit from a variety of perspectives. The target's market is usually adjacent to their own, and they can trust their informed impressions of how the target's organization and processes stack up.

Private equity firms (and later-stage VCs) are different; as financial buyers, they may not have technical resources on staff. If they are funding a portfolio company to acquire another, the company's technical staff will participate as described. Larger firms may have technical operating partners, but they also hire outside consultants for process due diligence.

A best practice is to start at a high level and drill down into areas of concern, either based on importance to the investment thesis or on the need to answer high-level questions. Of course, the need to prioritize entails deprioritizing some areas. The due diligence team can divide and conquer by assigning multiple people to respective areas. But beware of overtaxing the target's resources.

Lastly, insights also come from personnel below senior management—they are more likely to know and tell it like it is. It's not always possible, however, because the people further down in the organization are often not deal-aware. If speaking directly to less-senior people is not possible, digging into documentation and process can give some sense of how well the organization aligns with the senior management–eye view.

Internal documentation (or lack thereof) can be very telling. A well-organized target should not have trouble producing it. Judge the quality of processes relative to the size of the organization, and don't necessarily judge a small company by big company standards. At the same time, be mindful of what's on paper. For one thing, the ink may not be dry yet. That is, it's not unheard of for a company to quickly define and codify processes to try to look good in diligence. It's worth asking how long processes and documentation have been in place. More commonly, documentation may be out-of-date or never really implemented. Many policies and processes sound good but only sit on employees' desks without ever actually becoming part of the organizational fabric. It's worth digging into how employees are trained and how processes are monitored.

Evaluating the code itself

Experience reveals that senior management in target companies often don't have a great handle on issues in their own software. Evaluating a target's code is the only way to discover the accumulation of technical debt in it and get a high-level understanding of the cleanup that will be required. Software is never "done," and every codebase contains issues that need to be addressed going forward. It's the nature of the business, and expectations should not be for perfection.

However, most acquirers want to make sure that the issues are not excessive and that the code is constructed in line with industry expectations. And they want to ensure that they are properly accounting for the work required to address areas of concern.

The challenge is that the sellers do not want to release their code to the potential acquirer. The industry norm is for the target to display examples on a screen for the acquiring CTO to glimpse, but not to blithely hand over the codebase. And although a few curated samples may provide some insight, only deep and broad analysis of the code really paints the complete picture. As such, full code analysis can only be practically performed by a third-party trusted by both buyer and seller. Even this is a leap of faith for the seller, which is why a reputable third party is crucial.

The third party is able to delve into the software's quality, security, and composition. To comprehensively evaluate these aspects of software requires a combination of automated tools and expert curation of the output.

Assessing quality

There are two critical aspects of software assessment: design quality and code quality.

Design quality is the measure of how well the architecture can support the evolution of the software going forward. Does the structure of the code lend itself to future development for function and scale? How maintainable is the code? A well-designed software system will conform to modern distributed computing designs. It should be componentized, modular, and container-oriented to add to the ease with which it can incorporate a cloud-first strategy.

At first blush, it might seem that a review of design documentation would be the best approach. But the reality is that few organizations have the discipline to ensure that the structure of the code conforms to what's on paper, and conversely, to also ensure that what's on paper stays in sync with architectural system code changes. So it's vital to evaluate the design of the code.

Software can be well-designed but poorly written. Code should be extensively reviewed by an expert in software architecture. Bugginess is the attribute that is easiest to understand, but other aspects are important as well. Code written in languages appropriate to its intended function, server environment, or implementation is more efficient to maintain and scale, executes faster, and consumes less resources, all of which contributes to better performance and durability.

Also important is how comprehensible the code is. Software that is well-documented and written is easier for others to improve. Depending on the structure of the deal, this measure of transferability could be very important, such as in the case of an asset purchase with only temporary access to the developers.

Assessing security

Software security is a C-suite concern in any business and an important area of risk to be evaluated in every M&A transaction. Application security is the last line of defense and it's particularly important for public-facing applications. Internal threats are real too, but generally, security due diligence should be prioritized for software with an exposed attack surface.

In cybersecurity, the concept of "defense in depth" means that the best defense against attack is multilayered. Analogously, the best way to assess software security is by using multiple approaches to detect security bugs. No single technique finds every kind of flaw, especially within a constrained due diligence timeline. There are three useful techniques: A secure controls design review (the top-down view), static application security testing (the inside-out view), and penetration testing (the outside-in view).

A design review requires an expert to have access to the security architect responsible for the applications. The security architect can walk the expert through the controls in the design—for example, encryption, authentication, and password management. This type of assessment will determine whether the intent was to build appropriate security into the applications.

Static application security testing requires access to source code. It uses sophisticated tools that parse the logic of the code, applying rules to locate bugs that a hacker might take advantage of. However, the state of such tools is such that even the best do not produce 100% reliable results, so review by an application security analyst is also used. Sophisticated targets may use these tools themselves, but caveat emptor—an expert third party removes any concern of partiality.

Penetration (pen) testing requires access to running systems or test systems. It involves an authorized professional trying to hack in as a threat actor would. It is not unusual for targets to have hired third parties to perform pen tests in advance of due diligence. If the testing was performed by a reputable third party within, say, six months, it would not be unreasonable to accept the results.

Although an acquirer should expect to review the results, it should not expect its service providers to disclose the details about how to hack into the target's systems. Those details can be invaluable for remediation though, so arrange to gain access after the close.

Assessing open source and third-party software

Open source software is an important part of a target's security story, but it has the potential to contain legal risks as well. Typically, 75% or more of a target company's proprietary code is made up of open source and other third-party components. With literally millions of such useful components freely available on the internet, it's understandable that developers would avail themselves instead of reinventing the wheel. However, the magnitude of its use still surprises most people. And "most people" generally includes senior management in target companies.

The reality is that most companies, particularly smaller ones, don't have the policies and processes in place to track their developers' use of open source. Knowing what open source and other third-party components are in the code is required in order to assess associated risks. As such, questions about the open source code in a company's software are part of due diligence in almost every tech deal. But given that most companies can't accurately answer the questions, a third-party open source audit is very commonly called for—particularly if the target doesn't have solid answers to questions about how it manages open source.

Audits start by accurately and completely establishing which components are in the code. Once that's known, it's somewhat straightforward to identify license incompatibilities and known software vulnerabilities for each component. However, since there may be thousands of components in the codebases in a transaction, it helps to prioritize components for legal and application security review.

If the target has well-established policies and processes, and perhaps uses open source management tools, that should reduce concerns. However, human skill is needed to configure and run the tools, so be mindful of seller-supplied reports. Further, tools are limited, particularly when it comes to license issues. In high-risk scenarios like M&A, it's worth considering an expert audit, at least on software applications that are of most concern. It is also well worth involving open source–savvy IP counsel to sort through and advise on problematic licenses.

Conclusion

Going into software due diligence, acquirers should have a general plan but be nimble enough to adjust to the particulars of the investment as well as new information gained along the way.

Understanding the target's software development organization and processes is crucial to getting an idea of how future development will go. This is typically accomplished through document disclosures and discussions between technical peers. Strategic acquirers utilize their own staff. Financial buyers may bring in trusted consultants.

However, as important as it is to evaluate the code, buyers must also look back at development to date and assess the extent and type of technical debt that they will be inheriting. Any software contains risks—quality of architecture and code, security, and license issues with open source and other third-party components. Each type of risk could impact the deal and work plans going forward, depending on the particulars of the scenario and investment thesis.

Software due diligence is critical to informing tech deals and postclose integration. Well-planned due diligence can be the difference between an ultimately successful acquisition and buyer's remorse.

About Black Duck

Black Duck® offers the most comprehensive, powerful, and trusted portfolio of application security solutions in the industry. We have an unmatched track record of helping organizations around the world secure their software quickly, integrate security efficiently in their development environments, and safely innovate with new technologies. As the recognized leaders, experts, and innovators in software security, Black Duck has everything you need to build trust in your software. Learn more at www.blackduck.com.