GROWTH ACCELERATION PARTNERS

Understanding SOUP:

An Expert's Take on Risks, Tools and Best Practices

Posted on Growthaccelerationpartners.com on May 16, 2025

LEGACY APPLICATION MODERNIZATION

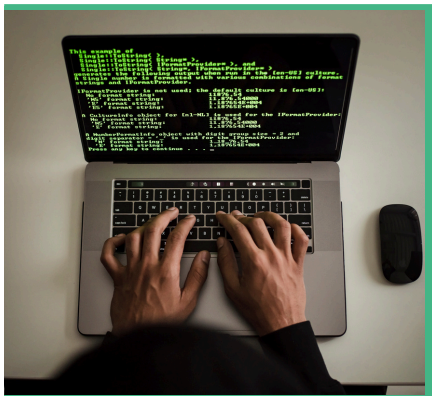
Using SOUP in your software is like building a house with bricks of unknown origin: some might be solid, others could crumble under stress, and you won't know which until the walls start cracking. In construction, every material used must meet safety and quality standards. Imagine receiving a pallet of bricks without labels, supplier info or proof they've passed inspection. You could build with them, sure, but you'd be betting the integrity of the entire structure on blind trust.

Similarly, SOUP refers to software components, libraries or code snippets whose origin, authorship and reliability are unclear. They might work for now, but if they haven't been vetted, they could introduce instability, security risks or compliance failures later on — when it's much harder (and costlier) to fix.

In highly regulated industries like healthcare, finance and defense, using unvetted software — especially components developed outside your organization — introduces critical risks. In this interview, I sat down with Ron Langer — a seasoned Modernization Advisor expert with over four decades of experience — to explore what SOUP is, why it matters, and how engineering teams can manage it without slowing down innovation.

So... What is SOUP?

The IEC 62304 standard — widely recognized in the medical software industry — defines **Software of Unknown Provenance (SOUP)** as:



"A software item that is already developed and generally available and that has not been developed for the purpose of being incorporated into the MEDICAL DEVICE (also known as 'off-the-shelf software'), or SOFTWARE ITEM previously developed for which adequate records of the development PROCESSES are not available."

Note: It's important to note that **SOUP is not exactly synonymous with the FDA's concept of Off-The-Shelf (OTS) software.** While both refer to external software not developed in-house, OTS is more of a legal-regulatory classification used by the FDA, whereas SOUP, as defined in IEC 62304, specifically captures the uncertainty around the development process and documentation — or lack thereof.

As Ron Langer explains it in practical terms, **SOUP encompasses any software component embedded in your application that your team neither developed nor maintains.** This includes open-source libraries, proprietary third-party tools, or licensed plug-ins. Once you integrate them, these components essentially become a "black box." You don't know how they were built, what quality controls they underwent, or how they may behave under stress — and that's where the risk begins.

In **regulated industries** like healthcare, this becomes a high-stakes issue. For instance, in medical software, the FDA mandates detailed documentation for every piece of software included in a device — even if it's a third-party library. If you can't demonstrate where a component came from, how it's maintained, and whether it's secure, your product risks being rejected or delayed in the approval pipeline.

So, why is SOUP drawing more attention today than ever before? According to Ron, the answer is clear: **security.**

"If you're using software that was written ten years ago," he warns, "it's likely not aligned with today's best practices for security."

*Outdated libraries often contain **known vulnerabilities** that hackers actively exploit. And legacy code — especially when unmaintained — becomes an easy entry point. Attackers don't need to breach your proprietary code; they'll look for the cracks in what you've borrowed.*

Consider a scenario in healthcare. If a third-party module you're using has a vulnerability that allows unauthorized access, attackers could gain entry to **protected patient health data** — information governed by strict laws like HIPAA. Even if no data is stolen, the fact that the system was compromised is enough to cause **serious disruption, reputational damage and regulatory consequences.**

Yet, despite these risks, many companies still get SOUP wrong — mainly by **ignoring it altogether.**

Ron notes that teams often build applications using legacy components and forget to revisit them. Some are even running on frameworks that were deprecated over a decade ago — like VB6, which Microsoft stopped supporting in **2008**. What's worse, many of these third-party vendors have gone out of business, meaning the libraries haven't been updated — or secured — in over 15 years.

"That's a time bomb," Ron cautions. "If you don't regularly audit and upgrade your dependencies, especially in regulated markets, you're walking a very thin line."

Ultimately, SOUP isn't just a technical concern — it's a **business risk.** Treating third-party code as a set-it-and-forget-it solution creates vulnerabilities that grow with every passing year. The lesson? If you didn't build it — **you must verify it, document it and keep it current.**

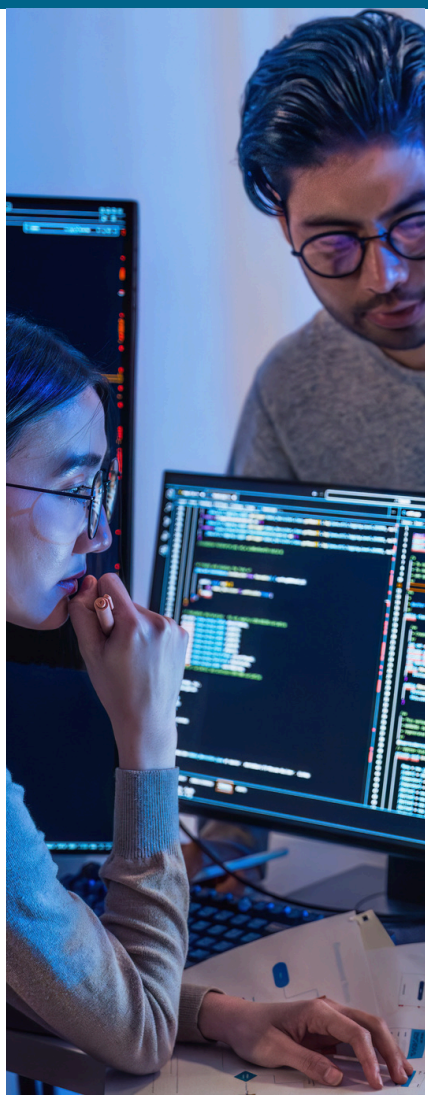
Best Practices for Identifying and Managing SOUP

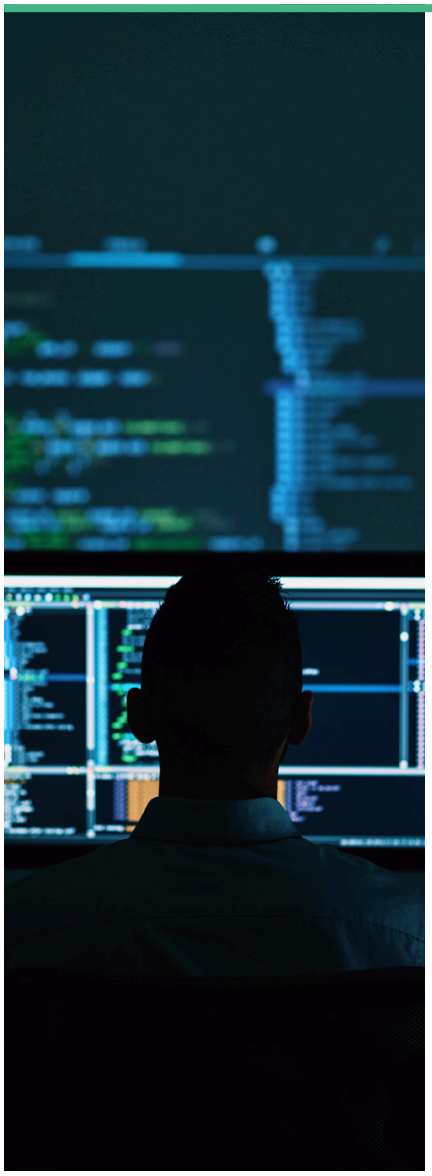
It Starts With Visibility

Start by scanning your codebase. SAST tools can help detect external libraries and identify outbound calls. However, automated tools often have limitations, meaning you'll also need to manually analyze the source code to capture the full picture. Then evaluate:

- Is the version still supported? And if it's not, then what do we do about it?
- Has it been updated recently (ideally in the past 12 months)?
- Is there a safer or better-supported alternative?
- Has it been periodically (once a month or quarter) scanned and evaluated for any Critical Vulnerabilities and Exposures (CVE)?

Using third-party software isn't inherently bad — it just needs to be treated like any other asset in your system. That means applying the same rigor to maintaining it as you would your own code.





Don’t Skip Legal Due Diligence

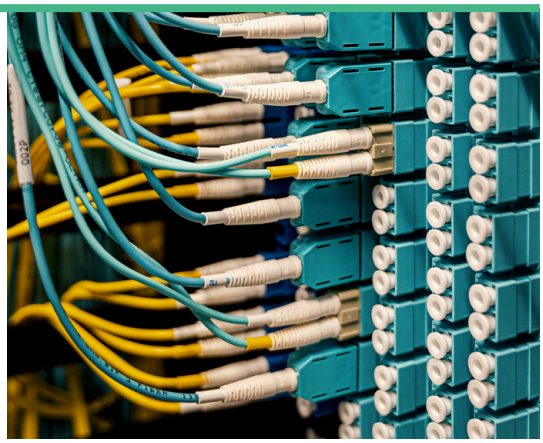
Equally important are the **legal implications**. Every piece of third-party software comes with a license, and not all licenses are created equal. Open-source licenses, in particular, vary in terms of how software can be used, modified and redistributed.

Some licenses allow unrestricted use and changes, while others impose strict conditions, such as requiring that any modifications be open-sourced or that commercial use is restricted. Before integrating third-party code into any commercial product, **get legal approval** and confirm that the license aligns with your business model and compliance requirements.

Additionally, **don’t overlook strategic licensing risks.** There’s a growing trend of popular open-source projects shifting their license models — either closing their source entirely or introducing a paid-for-commercial-use clause. Major names like MySQL, Terraform and Redis have made such changes, and we’ve seen others heading in the same direction. In one client project, we identified similar patterns in common OSS libraries like FluentAssertions, AutoMapper and Mediatr. It is all about doing strategic analysis and understanding the risk profile for any dependency.

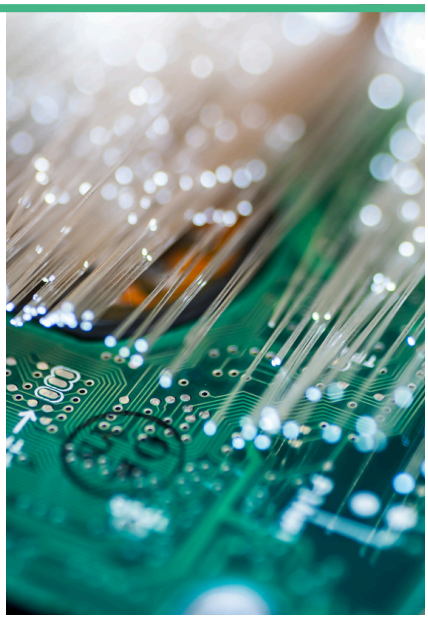
Balancing Risk and Speed

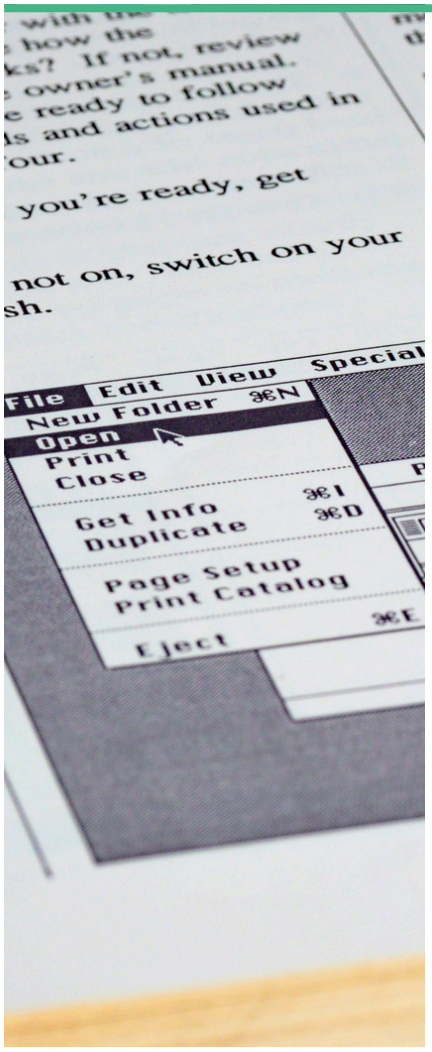
So, how can engineering leaders manage SOUP **without slowing innovation?** It’s all about finding the right balance. For example, if you need a PDF viewer, don’t reinvent the wheel — but choose a solution that’s **actively supported by a reputable vendor.** The key is to prioritize **mature, well-maintained tools.**



If you're going to use open-source components, Ron suggests leaning on those backed by **recognized organizations** like Microsoft or Google. “Blazor is open-source but maintained by Microsoft. Angular is maintained by Google. These aren't one-off GitHub projects that someone uploaded five years ago and then abandoned. They're backed by real teams and release cycles.”

In contrast, relying on obscure, single-developer projects — no matter how clever — introduces long-term risk if the codebase goes stale or the developer disappears. **The sustainability of the software matters as much as functionality.**





Document Everything

Finally, don’t overlook the importance of **comprehensive documentation**. Keeping a clear record of all SOUP components — including version numbers, licensing info, update history and usage — is not just a compliance requirement.

“Documentation makes your stack transparent,” Ron notes. “It helps engineering teams stay aligned and it gives legal and business leaders visibility into potential risks.” In other words, **you can’t manage what you can’t see**, and without documentation, SOUP becomes a silent liability.

But beyond written documentation, there are standards around **Software Bill of Materials (SBOM)** that enable richer tracking of dependency metadata, such as source, version, license and file hashes to verify authenticity. In regulated industries — especially healthcare — a SBOM in the Software Package Data Exchange (SPDX) is increasingly required, including by the FDA. Fortunately, vendor and language-specific tools can automate much of this process — like Microsoft’s SBOM tool for .NET applications.

REAL-WORLD APPLICATION

When it comes to tackling **Software of Unknown Provenance (SOUP),** GAP brings two critical advantages: **experience and process.**

Most organizations simply don't have the internal resources — or the time — to conduct full audits or modernize outdated components. It's not something they do every day. That's where we step in. With specialized expertise, we help identify aging or unsupported software dependencies, recommend targeted upgrades and execute migrations without disrupting our clients' core development activities.

Sometimes, the issue isn't just technical complexity; it's about **capacity**. Managing SOUP doesn't always require deep domain knowledge of the client's application itself. What it does require is rigorous migration practices and the ability to methodically triage risks across large, interdependent codebases.

Recently, we partnered with a **leading medical equipment manufacturer** to conduct a comprehensive evaluation of their software systems. Their stack included a number of third-party components — some of which were **multiple versions behind** — creating potential security vulnerabilities and compliance risks.

We performed a **full audit** of their third-party libraries, assessed compatibility with modern frameworks and designed a strategic roadmap for upgrades, replacements or deprecations where needed.

Our evaluation process covered:

- Identifying current versus latest available versions
- Ensuring compatibility with **.NET Framework 4.8**
- Reviewing licensing and legal considerations
- Analyzing migration complexities and proposing alternative solutions
- Developing potential long-term maintenance strategies

Through this initiative, we delivered two major benefits:

1. **Improved Security:**

   By upgrading to supported and actively maintained software versions, the client reduced their exposure to known vulnerabilities and strengthened the overall resilience of their platform.

2. **Improved Security:**

   In regulated industries, products cannot be released without demonstrating control and validation over every third-party component. Our work enabled the client to meet these critical regulatory requirements efficiently.

3. **Reduced Dependency Risk and Lowered TCO:mproved Security:**

   As a third-party modernization partner, we bring a fresh perspective. During the engagement, we identified opportunities to eliminate unnecessary third-party libraries — migrating to built-in platform capabilities that are better supported and easier to maintain. For example, we replaced standalone components like TopShelf with native Windows services and substituted SharpZipLib with libraries already included in the .NET Framework. This reduces the long-term maintenance burden and improves licensing clarity, ultimately lowering the total cost of ownership.

Beyond immediate upgrades, our proactive approach provided the client with a clear strategy for future maintenance and compliance. This ensured that their platform remains **secure, high-performing and aligned with regulatory expectations**, reinforcing their commitment to delivering **safe, high-quality products** in a complex, regulated market.

Final Thoughts

SOUP isn't just a regulatory headache — it's a risk multiplier. But with the right approach, tools and partnerships, it can be managed proactively. The key? Treat every third-party component like a structural element in your product's foundation. Because in software, as in construction, quality always matters — especially when you can't see the cracks until it's too late.

Have a project in mind we can help with? Don't hesitate to contact us.

About the Expert

Ron Langer holds a degree in Computer Science and Mathematics. With over 40 years of experience — including 13 years at IBM — he specializes in software modernization, legacy system upgrades and compliance-focused development. From mainframes to mobile apps, Ron has helped clients across industries navigate complex software transitions.

ABOUT GROWTH ACCELERATION PARTNERS:

At GAP, we consult, design, build and modernize revenue-generating software and data engineering solutions for clients. With modernization services and AI tools, we help businesses achieve a competitive advantage through technology.

To find out more, please visit WeAreGAP.com

company/growth-acceleration-partners/ @GrowthAccelerationPartners @GAPapps