It's Time to Rethink Our Enterprise Software Licensing Practices

By

Steven R. Kursh, Ph.D., CSDP, CLP
Software Analysis Group, Cambridge, MA,
s.kursh@softwareanalysisgroup.com

Pratik Patel, BS, MBA
Software Analysis Group, Cambridge. MA,
p.patel@softwareanalysisgroup.com

ABSTRACT

Our objective in this paper is to discuss how three advancements in the enterprise software industry: 1) Software as a Service; 2) End of Availability; and 3) Software in Suites have impacted enterprise software licensing practices. Taken together, these three advancements reduce customer control over version lifecycle and purchasing scope while also shifting more standardization and timing decisions to the software vendor. These advances, particularly in combination, have impacted the acquisition process for enterprise software by increasing short-term costs for companies while also providing many benefits to enterprise software customers. We conclude that those of us involved with licensing software at enterprises need to rethink our licensing practices.

Introduction

Advances in technology and business practices are requiring those of us involved with licensing enterprise software to adjust to a new reality. Three related advances are worth noting. First, licensing has shifted toward subscription-based access, most commonly delivered as Software-as-a-Service or SaaS, instead of perpetual licenses that customers deploy and operate in their own environments (on premises or customer-managed hosting, including public-cloud infrastructure).[1] Second, vendors are increasingly implementing and enforcing End of Availability (EOA) and End of Support policies that limit how long perpetually-licensed versions remain available for purchase, maintenance, and updates. Third, vendors are packaging software programs as integrated suites sold by subscription, which reduces the ability for enterprises to license only one discrete program to meet narrow needs.

Taken together, these advancements reduce customer control over version lifecycle and purchasing scope while shifting more standardization and timing decisions to the software vendor. More specifically, these advancements, particularly in combination, have impacted the acquisition process for enterprise software by increasing short-term costs for enterprise customers while providing benefits in reduced business and security risks, more robust functionality, and increased productivity.

Our objective in this paper is to review these major advancements and to discuss how these advancements may impact your work as a licensing professional. We begin with the shift to SaaS because it has enabled the other two advancements. We then address EOA as the lifecycle consequence of version sprawl and finally we then discuss suites as the packaging and commercial structure that is increasingly part of the software licensing.

The SaaS Advancement

During the past several years software companies and their enterprise company customers have shifted to SaaS as the primary way to access and use software.[2] Although companies and government agencies have used computers for over seventy years, the development and use of SaaS for enterprises began about twenty-five years ago with NetSuite, an accounting and financial management SaaS targeting SMB (small and midsized businesses) and Salesforce's CRM (customer relationship management). With the dotcom crash in the year 2000 along with other factors it took several years for SaaS to become readily acceptable as an alternative to on premises deployment of applications by most enterprises and governments.

1 Henceforth we will refer to these "own operations" as "on premises."

2 We discuss how SaaS has disrupted the enterprise software industry in a recent article. See, Kursh, Steven and Pratik Patel, "Assessing the Impact of Software As A Service (SaaS) Innovations on Disrupting the Enterprise Software Industry," *The Business Education Innovation Journal*, Vol. 16. 2, December 2024.

SaaS is now the primary method for selling rights to use software.[3] Many enterprise software companies no longer license software for on premise provisioning and instead sell SaaS subscriptions.[4] Most of these enterprise software companies still provide support and maintenance for existing on premise licensees due to some licensees preferring not to make the switch to SaaS due to financial, inertia, security, and regulatory factors, among other factors.

IDC, a leading IT-research company, forecasts that spending by companies for SaaS and ASP[5] accounted for over 70 percent of the enterprise software market as of 2024. Another leading research organization, Gartner, found similar growth numbers for SaaS.[6]

Clearly, there are a wide and deep range of categories of SaaS products available to companies and government agencies, particularly compared to licenses for software provisioned on premises as a product option. Based on our experience this wide and deep range of categories far exceeds what was ever available with on premises deployment of licensed enterprise software.

Obviously, no one single factor explains the advancement of SaaS. Some obvious factors are technology, economics, and business design (*i.e.*, models).

A key difference between SaaS and traditional on premises deployment is that SaaS applications are designed to be hosted in the cloud by the software vendors versus by the enterprise software customers in their own-controlled facilities. A SaaS vendor hosts its application(s) with cloud service providers like Microsoft's Azure, Amazon's AWS, IBM Cloud, Google Cloud, Rackspace, or other hosting services. Alternatively, a SaaS vendor can host its application(s) on its own cloud, which we have seen particularly prominent in vertical markets where security, regulatory, and unique business factors may be relevant. Some cloud services vendors, for example, IBM, offer a hybrid cloud option (a combination of a public cloud and private cloud) in response to these needs from enterprises and governments.

The technology benefits of SaaS are that a user at an enterprise, assuming s/he has rights to use the SaaS application, may access the application from a web browser. This includes not just desktop computers, but also mobile devices, *i.e.*, phones and tablets. In effect, user interfaces (UI) and, more broadly, user experiences (UX) with SaaS facilitates training time of personnel. Almost all of us are comfortable with using SaaS applications like Gmail, Quicken, and apps on our mobile devices.

These personal experiences enable most of us to learn and adopt SaaS applications relatively easily and certainly faster than what many people did in the past with most perpetually-licensed software that was sold for the customer to provision on their servers. From the perspective of IT managers and C-level personnel, this ease of use and shorter learning curve often lead to faster implementations, which can make SaaS more attractive even when SaaS applications are less customizable or flexible than perpetual licensed software.

Focusing on economics, SaaS has cost advantages since an enterprise does not need to invest in large upfront costs for hardware, networking, and other critical infrastructure, including even physical facilities, thus reducing the TCO (total cost of ownership). In our experience enterprises also mitigate many of the implementation failure risks and costs associated with on premises deployment.

Another cost benefit with SaaS is that enterprises using the software do not need to have as many technical and support staff as they would when compared with a perpetual-licensed software. Consider alone the cost savings that favor SaaS such as no hassles with system implementation, no data backup, not having to install and test upgrades and patches, and, critically important for most enterprises, security and data protection.

In fact, one research firm found that TCO with SaaS could be reduced by over 70 percent as compared with on premise deployments. Even if this finding of an over-70-percent reduction is off by half, a cut in TCO by 35 percent is significant, particularly in today's environment where IT management are continually asked to do more with less.

Additionally, from the perspective of an enterprise facing peak demand for its software, say, for example, a retailer that needs to process transactions at peak periods such as the holiday shopping season, SaaS provides elasticity because they are typically built on top of IaaS.[7] In other words, the software capability can quickly scale with increases in demand during peak periods and with growth in subscribers overall.

This elasticity enables scale and significantly decreases the investment costs for subscribers since there is no need to acquire expensive hardware with the capability of handling future growth demand

3 Datanami, "Gartner Forecasts Worldwide Public Cloud End-User Spending to Surpass $675B in 2024." https://www.datanami.com/this-just-in/gartner-forecasts-worldwide-public-cloud-end-user-spending-to-surpass-675b-in-2024/. August 5, 2024.

4 Lin, B, "SAP share surge shows companies' cloud strategies are alive and well in AI boom." *The Wall Street Journal.* July 26, 2024.

5 ASP definition by Gartner: "An application service provider (ASP) is defined as an enterprise that delivers application functionality and associated services across a network to multiple customers using a rental or usage-based transaction-pricing model." https://www.gartner.com/en/information-technology/glossary/asp-application-service-provider? February 21, 2026.

6 Datanami, 2024.

7 IaaS definition by by Gartner: "Infrastructure as a service (IaaS) is a standardized, highly automated offering in which computing resources owned by a service provider, complemented by storage and networking capabilities, are offered to customers on demand. Resources are scalable and elastic in near real time and metered by use." https://www.gartner.com/en/information-technology/glossary/infrastructure-as-a-service-iaas. February 21, 2026.

needs. It also reduces, importantly, transaction costs. When demand peaks the software scales immediately.

In our experience the faster positive returns help SaaS applications to enable success stories and build credibility among an enterprise's users. Just as an individual can quickly set up a Gmail and Dropbox account without installing software on their own device, an enterprise can often get up and running relatively quickly with a SaaS application. This agility enables an overall positive experience that builds momentum and enhances the software company's credibility.

Another economic benefit enabled by the advancement of SaaS solutions is that the process and related costs for rolling out new features and fixing software bugs is often easier, less risky, and significantly less costly as compared with the past. In contrast to software running on premises, a SaaS company monitors the real-time use of its software, detects and resolves issues, and makes incremental changes that can be easily assimilated by subscribers. In turn, the SaaS solution becomes more "locked in" with many subscribers. More importantly, by controlling the continuous update lifecycle process, SaaS companies can innovate continually, creating a faster and more efficient innovation cycle.

Another benefit, referenced above, provided by SaaS is better security. Although each of us should have security software installed and operational on our computers and other devices, security-related risks are lower with software that runs in the cloud at vendors like AWS Microsoft Azure, Google Cloud and others as compared with an organization obtaining, installing, and updating security software on its own hardware.

This is largely because cloud vendors can apply security patches and configuration updates centrally and quickly, instead of relying on each organization to maintain consistently every server and endpoint on its own. The cloud vendors also have dedicated security teams, continuous monitoring, and built-in redundancy at a scale that most organizations cannot reasonably replicate internally. This reduces the likelihood that a known vulnerability stays unpatched long enough to be exploited.

The reality is that no software application can be entirely free of security risks, but having a SaaS application is less risky than if the software runs on premises.[8] It is also more economical for SaaS application providers to utilize security solutions offered by their IaaS (Infrastructure as a Service) provider (*i.e.*, the cloud vendor) instead of trying to do it themselves, which allows them to concentrate on building and running software applications.

In sum, SaaS came about with the internet and, through a series of technology and business innovations (discussed below), has effectively replaced traditional, customer-deployed software for most users, whether enterprises, government or individuals. The technology innovations have enabled software companies to provide a superior product at a lower cost, with lower risk, better features and better performance. SaaS has effectively "set the table" for more advancements and, accordingly, is changing practices in software licensing.

The EOA Advancement

In our experience the primary driver of EOA (End of Availability) is the accompanying problems with software that has aged, despite on the surface being operational at many companies.

While some of us may think that EOA reflects primarily greed from enterprise software companies, the fact is that many enterprise software applications were developed and licensed when IT infrastructures and software development tools were quite different than what exists today. These older software applications have become increasingly difficult to maintain, support and update due to factors such as technical debt, obsolete code, reduced functionality and other constraints.[9] In addition, organizations often struggle to recruit and retain the talent needed to maintain antiquated code.

Some history worth reviewing is that for much of the software industry's history, enterprise applications were sold as stand-alone products rather than as integrated suites. Hence, most companies had many software applications largely running independently of each other. Even if two pieces of software were from the same vendor, they could have different architectures, user interfaces, compatibility, *etc*. because the software was developed over different time periods and different teams.

Today many companies operate on different combinations and versions of a vendor's software, often customized or configured uniquely for each company's business needs and environments. Over time, version growth occurred, *i.e.*, moving from version 1.0 to 2.0, to 2.01, to 2.11 and beyond, requiring support and maintenance across multiple generations.

Many enterprises still choose not to update, even when new versions are available, due to various operational and organizational constraints. In our experience companies often skip modernization or

8 Kursh, Steven and Pratik Patel, "Revisiting Goldilocks: Reasonable Measures to Protect Trade Secrets in an Era of Enterprise Collaboration and Cybersecurity Risk Management," *les Nouvelles,* Volume LX, No. 3, September 2025.

9 Gutteridge, Lance, "Enterprise Software Is the Hardest Software to Write," *Codeburst*, August 24, 2018, https://codeburst.io/enterprise-software-is-the-hardest-software-to-write-c76d59725f3.

delay migration to a new version unless a cost-saving case is obvious or when there is a major operational, security, or business risk. Such decisions though may ultimately undermine organizational operations, security and strategic agility.[10]

Consider, moreover, the perspective of enterprise software vendors. By licensees postponing installation of upgrades, vendors eventually face an exponentially growing number of combinations to support. Compatibility testing, ensuring that software works properly across different hardware, operating systems, browsers, databases and mobile devices becomes a massive challenge. Even minor updates can trigger failures across enterprise environments due to variations in operating systems, middleware, or integrated third-party tools.

In effect, "the math doesn't work." For example, assume that an enterprise software vendor offers ten applications, each with five versions in the field. If so, the number of combinations becomes unmanageable. Mathematically, the total number of permutations is: 5^10. This equals 9,765,625 (about 10 million) possible combinations.

In other words, there are nearly 10 million possible version interactions that a software company's staff must account for if its enterprise licensees are free to use any mix of the software applications and versions. Maintaining, supporting and updating every variant of every version is extremely difficult in practice, requiring compatibility testing that grows exponentially with each new release and unique combination. Of course, a software vendor may have fewer than ten applications and fewer than five versions for each application, but the overall number of combinations can still be insurmountable from the perspective of ongoing support and maintenance for the enterprise software vendor.

Consider, too, that a recent survey from Saritasa found that 60 percent of organizations continue to run legacy systems, citing security risks and integration challenges as primary barriers to modernization.[11] These organizations simply have too many day-to-day fires to put out, as well as ongoing constraints on spending for IT. They also obviously face multiple demands for resources, not just IT-related needs. The "if something works, don't fix it" strategy has become a hallmark given the increasing demands on senior-IT personnel and departments. It's not surprising to us to learn that an enterprise client is using a software application that was originally licensed decades ago. Some clients, moreover, have stopped maintenance and updates. Several clients, while continuing to pay for maintenance and updates with a "just in case mindset," rarely install most updates from their software vendors except in extreme circumstances.

This approach has produced fragmented software environments that restrict vendors' ability to deliver consistent functionality, introduce new features quickly and address security vulnerabilities across all licensees. It also makes it far more difficult for enterprise software companies to leverage advances in modern hardware and operating systems. The result is an environment where both innovation and protection lag behind technology's pace, a reality that lays the groundwork for today's EOA with so many software applications.

Over time, enterprise software inevitably accumulates maintenance tasks, outdated code and architectural compromises that compound with each release. Technical debt grows as tools, software development kits (SDKs) and programming languages evolve. What was once stable, well-understood software becomes increasingly brittle, dependent on obsolete libraries and unsupported frameworks. The longer a software application persists without major modernization, the more fragile it becomes beneath the surface.

Additionally, outdated modules and languages introduce vulnerabilities that are difficult and, sometimes, practically impossible to patch without rewriting substantial portions of code. Refactoring, the process of restructuring software while preserving its functionality, becomes an engineering challenge when dependencies are deeply embedded and documentation no longer matches the underlying software and system.

In many cases, simply keeping old software and systems operational demands specialized knowledge that is no longer commonly known among developers working at companies that originally licensed the software. Indeed, in our work we often have been engaged by multinational and national enterprises that run applications written in COBOL, a language created in the late 1950s, where even routine security updates are extremely difficult to write because so few developers remain who are fluent in the software code.

Unlike smartphones, EVs and cloud-based applications that can update seamlessly, enterprise software operates within a complex and fragile ecosystem often across many organizations. Each update requires careful coordination across databases, integrations and infrastructure components that may differ from one customer to the next. Updates in enterprise environments can be unpredictable and when something goes wrong, the results can be catastrophic.

10 Mee, Paul and Chris DeBrusk, "Why End-of-Life IT Is Ruining Innovation and How to Fix It," Oliver Wyman, February 2024, https://www.oliverwyman.com/our-expertise/insights/2024/feb/eol-technology-how-to-avoid-risks-drive-digital-innovation.html.

11 Froehlich, Sabrina, "Legacy Software Modernization in 2025: Survey of 500+ U.S. IT Pros," Saritasa, August 25, 2025, https://www.saritasa.com/insights/legacy-software-modernization-in-2025-survey-of-500-u-s-it-pros.

Consider, by way of example, that in 2023 United Airlines had a system outage due to an update that forced it to halt departures nationwide on a busy travel day.[12] There are many other examples that are well known.

The reality is that once software reaches its end of life,[13] it stops receiving the updates, patches and bug fixes that defend against emerging threats. Each unpatched component effectively becomes an unlocked door in an organization's security perimeter. Fundamentally, we know that in enterprise computing, the greater the interconnectivity, the higher the consequences of failure.

Focusing on maintaining backward compatibility often forces software vendors to make difficult choices, including delaying or even skipping critical security hardening to ensure that their software, with the multiple combinations in the field, continues to function for licensees. Nearly every concession for backward compatibility reasons leaves licensees' software and, accordingly, the licensees' systems exposed to threats that legacy systems were never designed to withstand. Over time, this trade-off not only increases risks for licensees, it also erodes an enterprise software vendor's reputation. Any software update that "breaks" a legacy application can result in potential liability for the software vendor, even when the root cause is structural obsolescence or other factors driven by licensee decisions.

An August 2025 Microsoft update is an example of this situation. (Disclosure: one of the authors has previously worked as a consultant for Microsoft.) The update was intended to improve performance and security, but it broke backward compatibility with older enterprise applications. This event confirms the reality that modernization and preservation of legacy models with outdated applications are fundamentally at odds. Indeed, each new generation of software has a different architecture and fragmented legacy software will never be able to deliver equivalent functionality. This tension between progress and preservation lies at the heart of why EOA and standardized lifecycles have become engineering necessities rather than marketing choices.

Some people may think that EOA is more a marketing/sales issue driven by a desire by software vendors to increase revenue. While it is true that most software vendors are focused on growing revenue and profits, as, indeed, all companies on behalf of their stakeholders are, the reality is that EOA is grounded in well-established engineering principles and standards.[14] Ending support for legacy versions is not an act of abandonment driven by a desire to squeeze licensees for more revenue, but an engineering practice required to maintain security and stability while increasing functionality.

The Advancement to Suites

The third advancement that necessitates our rethinking about software licensing is enterprise software companies licensing software in suites. This change is not merely for commercial purposes, but often driven by technical necessity, for many of the same reasons we noted above when discussing EOA. SaaS architecture gives vendors total control over the environment, so updates can be tested once and rolled out in a far more predictable manner.

SaaS and other types of subscription models enable enterprise software companies to maintain uniform patching cycles, enforce a consistent security posture and deliver feature parity for all their users. By operating on a single, continuously updated version, enterprise software vendors reduce the complexity of supporting dozens of outdated releases and mitigate further cybersecurity risks inherent in fragmented product ecosystems by bundling them in integrated suites.

For vendors offering multiple complementary applications (*e.g.*, Microsoft Word, Excel and PowerPoint), the evolution toward integrated suites (*e.g.*, Microsoft Office 365) was a natural progression. Selling software as individual products creates overhead with compatibility testing, update management and cross-application performance. In contrast, suites come with centralized update management so security updates are coordinated across the full product line.

Adobe's 2013 transition to Creative Cloud marked a defining moment in this shift, demonstrating how ending perpetual licenses can standardize patching and accelerate delivery of new capabilities across all users.[15] (Disclosure: one of the authors has previously worked as a consultant for Adobe.) Similarly, VMware's 2024 EOA announcement for perpetual licenses underscored the industry's transition to cloud-based delivery with a unified subscription lifecycle.[16] (Disclosure: one of the authors has previously worked as a consultant for

12 Associated Press, "United Airlines Resumes Flights After Equipment Outage," *AP News*, September 5, 2023, https://apnews.com/article/united-airlines-flights-stopped-faa-ce9623ae0fd1e657830dc6acb6cccbc4.

13 The industry terms "end of life" and "end of availability" are generally equated.

14 In fact, EOA and suites are consistent with international standards such as ISO/IEC/IEEE 12207:2017 and 24728-1:2024. (International Standard ISO/IEC/IEEE 12207, "Systems and Software engineering – Software life cycle processes," Reference number ISO/IEC/IEEE 12207:2017). Also, International Standard ISO/IEC/IEEE 24748-1, "Systems and software engineering—Life cycle management—Part 1: Guidelines for life cycle management," Reference number ISO/IEC/IEEE 24748-1:2024 and International Standard ISO/IEC/IEEE 24748-2, "Systems and software engineering—Life cycle management - Part 2: Guidelines for the application of ISO/IEC/IEEE 15288 (system life cycle processes)," Reference number ISO/IEC/IEEE 24748-2:2024.

15 https://www.adobe.com/creativecloud/plans.html.

16 https://blogs.vmware.com/cloud-foundation/2024/01/22/vmware-end-of-availability-of-perpetual-licensing-and-saas-services/.

VMWare's prior owner EMC and as a consultant for VMWare's present owner Broadcom.) Many other companies including Archicad, Cisco and Microsoft illustrate this transformation by retiring perpetual licenses in favor of subscription models that ensure every customer operates within a consistent, modern and secure software environment.[17]

The benefits of this transition are tangible for end users and IT support teams at companies. A 2024 Nucleus Research report highlights that integration and consulting costs for standalone (best-of-breed) applications often exceeds the cost of the software itself, whereas integrated suites help lower technical complexity and support expenses, particularly for enterprises.[18]

Furthermore, subscriptions embed updates directly into the software lifecycle, removing the overhead of version management and reduce maintenance costs for subscribers. Subscription updates include security patches, feature enhancements and performance improvements that arrive seamlessly and create minimal disruption.

This approach has altered the relationship between enterprise software vendors and their enterprise customers from transactional to a continuous relationship and, accordingly, aligns interests around stability, innovation and long-term value. Software delivered via a subscription like SaaS uses modern development frameworks that enable vendors to build enhancements faster for their licensees because of the speed in which versions need to be released.[19] These changes have transformed software from a static purchase into an evolving service that strengthens the relationship between vendors and users where users expect stability and continuous innovation and vendors grow revenue through ongoing value delivery.

Going forward, we need to recognize that enterprise software vendors must embrace lifecycle discipline, unified architectures and continuous software delivery as the foundation of responsible engineering and best practices. The lifecycle approach recognizes that every product has a natural endpoint, *i.e.*, EOA and that expecting indefinite support is not feasible. Subscription-based models with unified architectures reduce version sprawl, simplify integration and create the cadence needed to deliver consistent updates that quickly push out new functionality and security fixes. Together, these practices form a viable roadmap for sustainable innovation that helps to enable value creation at subscribers.

The industry's shift towards cloud-based delivery with unified subscriptions in a suite is not merely modernization for convenience; it represents a scalable and disciplined path forward for enterprises. In a world of constant change and accelerating threats, the only sustainable strategy is rigorous lifecycle management. Software companies that treat EOA as an engineering obligation rather than as a business option, as well as offering suites if they sell multiple applications have defined and will continue to define enterprise software and will set the standards for customer expectations. Similarly, enterprise licensees need to recognize the reasons why their software vendors are moving from the legacy model of perpetual licenses, customer provisioned and stand-alone software applications. Accordingly, those of us involved with licensing software at our enterprises need to rethink our licensing practices. ■

17 https://www.mynewsdesk.com/graphisoft/pressreleases/graphisoft-announces-next-phase-of-its-shift-to-future-proof-subscription-model-3345644, https://www.cisco.com/c/en/us/products/collateral/unified-communications/unified-communications-licensing/eos-eol-notice-c51-744286.html and https://clientsfirst-us.com/blog/end-of-microsofts-perpetual-licensing-era.

18 Campbell, Ian. "The Shift from Best-of-Breed to Integrated Suites: Streamlining Costs and Enhancing AI Capabilities," *Nucleus Research*, March 26, 2024, https://nucleusresearch.com/the-shift-from-best-of-breed-to-integrated-suites-streamlining-costs-and-enhancing-ai-capabilities.

19 "Software Product End of Life and Why It Is Good for Your Business," *ECI Software Solutions*, August 11, 2020, https://www.ecisolutions.com/blog/software-product-end-of-life-and-why-it-is-good-for-your-business/.