International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V2I2P108
Eureka Vision Publication | Volume 2, Issue 2, 65-72, 2021

Original Article

The Evolution of Software Delivery and the Rise of Saas Solutions

Uday Kumar Reddy Gangula
Independent Researcher, USA.

Abstract - The paper studies software delivery model development from mainframe times to Software as a Service (SaaS) by analyzing technological advancements and development approaches alongside economic factors that transformed the industry. The analysis begins with the first era of tangible software that required on-premises deployment and the Waterfall methodology before discussing the internet disruption and the subsequent flawed Application Service Provider (ASP) model. The success of SaaS results from the union of mature cloud computing with multi-tenant architecture and Agile/DevOps practices. The paper investigates how SaaS transformed business models through its adoption of subscription-based recurring revenue instead of perpetual licensing while analyzing its impact on Total Cost of Ownership (TCO). The paper discusses ongoing security risks and vendor lock-in problems while studying upcoming trends that include verticalization and artificial intelligence integration.

Keywords - Software Delivery, Software as a Service (SaaS), Cloud Computing, On-premises, Application Service Provider (ASP), Waterfall Model, Agile, DevOps, Total Cost of Ownership (TCO)

1. Introduction

The development of software delivery methods, starting from CD-ROM-based shrink-wrapped products, has evolved into SaaS subscriptions, which users access through web browsers, marking a fundamental transformation in digital tool creation, distribution, and usage [1]. The evolution demonstrates both technological progress and changing economic conditions, as well as new software producer-user relationships. The paper indicates that SaaS developed naturally through multiple decades of technological advancements in computing infrastructure, networking capabilities, and software engineering practices.

This analysis traces the historical and technical development of SaaS in chronological order to explain its current dominance. The paper begins with a discussion of physical Software, which included mainframe computing systems and the product-based approach of the personal computer revolution. The paper examines how the internet enabled digital distribution possibilities while studying the brief Application Service Provider (ASP) model from the late 1990s. The paper explains why ASP failed while SaaS succeeded through discussions about essential factors, including cloud computing, multi-tenant architecture, and Agile and DevOps methodology integration. The paper examines the economic effects of SaaS on Total Cost of Ownership and the shift to recurring revenue models.

2. The Era of Tangible Software: From Punch Cards to Shrink-Wrapped Boxes

2.1. The Mainframe Paradigm and Centralized Computing (1950s-1970s)

The introduction of IBM System/360 marked the beginning of commercial computing through its centralized processing system that required specific hardware for software execution [2]. The development process through punch cards required extended time because manual operations and restricted machine availability created prolonged feedback cycles. The requirement for thorough initial planning resulted in the development of the linear phase-gated Waterfall methodology, which focused on creating error-free designs to prevent expensive post-development corrections [3].

2.2. The Personal Computer Revolution and the "Software-as-a-Product" (SaaP Model 1980s-1990s)

The Apple II and IBM PC launched a personal computer revolution, which separated software from mainframes to establish a broad market for standardized applications [4]. The "Software-as-a-Product" (SaaP) model emerged, with software sold as tangible goods on physical media like floppy disks and CDs. The distribution method required software releases at infrequent intervals with high stakes, which supported the Waterfall model’s requirement for detailed upfront planning to prevent costly post-release errors [4]. The extended development period made it challenging to respond effectively to user requirements and market shifts.

2.3. The Dominance of the Waterfall Methodology for Physical Distribution

The Waterfall model became official in 1956 and achieved standardization by 1985 through its structured development process, which included requirements followed by design, then implementation, testing, deployment, and maintenance [3]. The SaaP model required a rigid approach because it used physical distribution methods that needed a complete upfront design to prevent errors. The approach failed to recognize the software's flexible nature, which resulted in performance issues. The 2002 Standish Group study revealed that 64% of software features remained unused, which demonstrated how the model failed to meet user requirements because of its extended development time and insufficient feedback [3].

3. The Internet as a Catalyst for Change

3.1. The Dawn of Digital Distribution and the Client-Server Architecture (1990s)

The World Wide Web served as the primary driver that transformed software delivery methods—the web functions as a platform for distributing static HTML content during its initial development period. The web evolved into an interactive platform for dynamic applications through the implementation of JavaScript and CSS technologies. The technological advancement allowed software developers to deliver applications directly to users through internet downloads, which eliminated their dependence on physical distribution methods.

The client-server architecture emerged as the leading framework for web applications during this period. Users accessed the application logic and data through a centralized server, which operated as the powerful central point in this model. The first architecture shift toward centralized application management emerged as a fundamental principle that would eventually define cloud computing. The server-side code updates became instantly accessible to all clients through this revolutionary change, which ended the SaaP model's slow upgrade cycles. The Waterfall model's dominance faced its first major challenge because the cost of bug fixes and update deployments decreased dramatically.

The web developed through two essential paths, which led to the emergence of SaaS. The "Software-as-a-Resource" (SaaR) model became possible through the web because it delivered dynamic content to users, with the output being the only aspect they could access, as seen in search engines like Google. The business model operated through advertising revenue streams. The web functioned as an efficient platform to distribute downloadable on-premises applications, which improved the traditional Software-as-a-Product (SaaP) model. The combination of centralized processing with digital delivery systems created the base for SaaS, which united managed infrastructure with interactive user experiences.

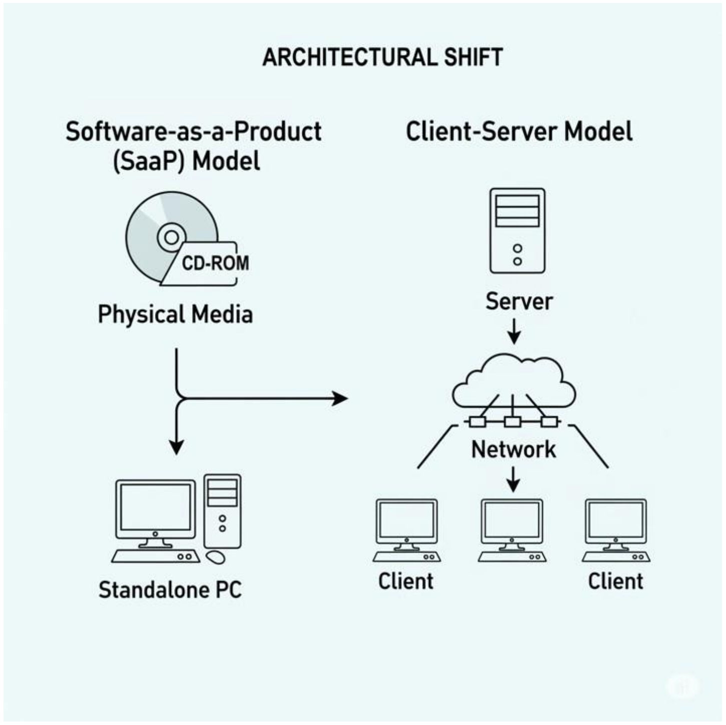

Figure 1. Illustration of the architectural shift from the "Software-as-a-Product" (SaaP) model to the client-server model

3.2. The Application Service Provider (ASP) Model: A Forerunner to SaaS (Late 1990s)

The internet commercialization during the late 1990s led to the emergence of Application Service Provider (ASP) as the initial attempt to market software through remote delivery services. [5] The data centers of ASPs operated as software application hosting facilities, which provided network-based access to customers through monthly subscription fees. The business model of modern SaaS operates under the same principles as ASPs by offering reduced software license expenses and eliminating server infrastructure needs and application management responsibilities. [5]

The model gained popularity among healthcare providers seeking to utilize complex applications, such as Electronic Health Records (EHRs), without incurring significant capital expenses. The ASP model operated with a different design structure than current SaaS solutions. Each customer received their own software instance on the provider's servers through the single-tenant design [6]. The ASPs adopted a "lift-and-shift" strategy, moving existing on-premises client-server applications into hosted environments rather than redesigning them for delivery service. The approach maintained the inefficient aspects of outdated systems by requiring manual customer setup, limited scalability, and requiring extensive labor for instance-specific upgrades and patches [6]. The service-based model of ASPs remained restricted by their outdated product-centric architecture.

3.3. An Autopsy of the ASP Model: Why a Good Idea Failed

The ASP model demonstrated promising potential but failed to achieve long-term success because numerous providers went out of business during the dot-com bust of 2000–2001. The model failed due to a combination of technical restrictions, unrefined business approaches, and unfavorable market elements. The single-tenant architecture of the model proved to be both inefficient and expensive in technical terms. The process of manual instance provisioning for each customer restricted scalability and eliminated potential cost savings [6]. The performance suffered due to inadequate internet infrastructure during the late 1990s, when users primarily relied on dial-up and early broadband connections that were unreliable. Users experienced poor performance when running complex applications through these networks. The hosted applications were not fully developed since they represented version 1.0 releases with basic features and stability issues, which made them unattractive to businesses using established on-premises systems.

The business model of ASPs faced challenges with pricing strategies and an insufficient customer base to maintain financial stability. Organizations refused to transfer their sensitive data to third parties because they viewed their networks as insecure [6]. The dot-com crash eliminated funding and market confidence, which led to the demise of ASPs that had technically sound solutions. The ASP model demonstrated premature innovation as its final result. The model correctly predicted the transition from software ownership to service-based delivery yet failed to establish the necessary supporting infrastructure. The architecture lacked multi-tenancy capabilities, while network infrastructure was insufficient, and pricing models were not developed; the market lacked trust in hosted services. SaaS achieved success many years after its inception because the essential technological, economic, and cultural foundations had reached maturity.

4. The Ascension of Software as a Service (Saas)

4.1. Foundational Pillars: The Role of Cloud Computing and Virtualization

The early 2000s established an ideal technological setting that allowed SaaS to develop. Virtualization and cloud computing emerged as the two essential innovations that established the basis for this model to become viable. The technology of virtualization, which VMware pioneered, allowed one physical server to operate multiple independent virtual machines that functioned as separate operating systems with their own applications. The innovation broke the conventional one-to-one hardware-software connection, which resulted in better resource management and operational efficiency.

Public cloud computing emerged as the second major advancement, which brought revolutionary changes to the industry. Amazon Web Services (AWS) launched its Infrastructure as a Service (IaaS) platform in 2006, which provided customers with on-demand programmatic access to computers, storage, and networking resources. Through abstraction, the data center became a utility that enabled Salesforce.com and other SaaS providers to concentrate on application development instead of managing hardware. The elimination of complex infrastructure operations through IaaS removed the major obstacle that prevented scalable service delivery.

The fundamental aspect of this revolution was abstraction. Virtualization separated the OS from hardware, while IaaS separated the entire data center. SaaS abstracted the application from the user, relieving them of installation, maintenance, and infrastructure concerns [1]. The abstraction layers progressively decreased complexity and costs, which formed a beneficial cycle that allowed SaaS to achieve effective scaling. The operational efficiencies and scalability of virtualization and IaaS made SaaS both economically and technically possible.

Table 1. Comparative Analysis of Software Delivery Models

Feature	On-Premise (SaaP)	Application Service Provider (ASP)	Software as a Service (SaaS)
Architecture	Client-Server, Monolithic	Single-Tenant (Hosted Client-Server)	Multi-Tenant
Primary Cost Model	Perpetual License (CapEx) + Annual Maintenance	Monthly/Annual Hosting Fee (OpEx)	Subscription Fee (OpEx)
Deployment	Local Installation via Physical Media/Download	Remote Access, often with some local components	Web Browser Access (No Local Install)
Maintenance/Updates	Customer-Managed	Provider-Managed (per instance)	Provider-Managed (single codebase for all)
Scalability	Requires new hardware/license purchase (Slow)	Limited by the provider's manual process (Slow)	Elastic, On-Demand (Rapid)
Hardware Ownership	Customer	Provider	Provider (often via IaaS)

4.2. Defining SaaS: Core Characteristics and the Multi-Tenant Architecture

The term "SaaS" emerged in the late 1990s, but cloud infrastructure maturity enabled its official definition and broad market adoption. The National Institute of Standards and Technology (NIST) defines SaaS as a model that allows consumers to access provider applications running on cloud infrastructure through thin clients such as web browsers. The consumer maintains no control over or responsibility for managing the underlying infrastructure [1].

Several core attributes define SaaS:

- **Subscription-Based Licensing:** Customers pay a recurring fee, replacing enormous upfront costs with predictable operational expenses.
- **Web-Based Access:** Applications are accessed via standard browsers, eliminating local installations and simplifying updates [1].
- **Centralized Management:** The provider handles all aspects of service delivery, including updates, security, and availability.

The technical innovation that defines SaaS is multi-tenancy because it distinguishes it from ASP [6]. A multi-tenant architecture enables one software instance to serve multiple customers (tenants) through shared infrastructure and codebase [7]. The logical separation of tenant data ensures both privacy and security [8].

The architectural advancement made SaaS economically feasible. A multi-tenant system allows developers to deploy updates and bug fixes once, so all users receive them instantly [9]. The shared infrastructure between customers reduces operational expenses, which enables providers to deliver complex software at affordable prices. The operational efficiency of SaaS companies allows them to provide enterprise-grade functionality to small and medium-sized businesses [10]. The SaaS model depends on multi-tenancy as its economic foundation.

4.3. The Agile and Devops Symbiosis: Methodologies for Continuous Delivery

The development methodologies that emerged simultaneously with multi-tenancy provided the necessary procedural structure to support service-based model requirements. The Waterfall method proved unsuitable for internet-delivered services because it maintained strict phases with rare releases.

The Agile Manifesto emerged in 2001 to present principles that emphasized iterative development and rapid feedback alongside cross-functional collaboration. Scrum and Extreme Programming (XP) provided development teams with the ability to deliver software through minor incremental releases instead of traditional long monolithic development cycles.

The delivery bottleneck remained unsolved by Agile implementation alone. The teams could develop new features at. A fast pace, yet the deployment process remained slow and manual with frequent errors. DevOps emerged to fill this gap through its purpose of uniting development teams with operations teams. The software lifecycle received special emphasis for automation through deployment pipelines under DevOps.

The essential practices of Continuous Integration (CI) and Continuous Delivery/Deployment (CD) emerged as critical elements because they automatically built and tested code after each commit and automatically deployed tested code to production [11]. The practices enabled SaaS providers to deliver updates both quickly and reliably, which supported their promise of continuous innovation.

The successful operation of SaaS depends on the combined power of Agile and DevOps methodologies. A “product” reaches completion after one delivery, but a “service” operates continuously through ongoing development while always remaining operational. Agile methodologies support fast and iterative feature development, yet DevOps provides the necessary safety measures for delivering features at regular intervals. SaaS established the business need for continuous delivery, which Agile and DevOps implemented as cultural and technical frameworks to achieve. These two elements create the operational base for the contemporary software service model.

5. The Transformational Impact of Sass

5.1. Economic Disruption: On-Premise vs SaaS Total Cost of Ownership (TCO)

The economic benefits of SaaS adoption stand as a primary reason for its adoption because they outperform traditional on-premises models according to research [12]. The deployment of on-premises software requires organizations to spend large amounts of money on perpetual software licenses, server hardware, networking equipment, storage solutions, and data center infrastructure.

The initial costs of deployment increase with ongoing operational expenses that continue throughout time. The ongoing expenses of software maintenance fees (18-22% of license costs), together with hardware replacement costs, power consumption, and most significantly, IT personnel salaries for environmental management and security [12]. Research from 2009 demonstrated that cloud-based solutions with 100 users produced total cost of ownership savings of 50% compared to on-premises solutions, primarily because of infrastructure and staffing expenses.

The SaaS model shifts capital expenditure into operational expenses through its subscription-based pricing structure [7]. The single recurring fee provided by providers includes software, maintenance, support, and upgrades, thus eliminating infrastructure costs and providing better budgeting predictability [5]. SaaS enables organizations to redirect their resources from operational costs to strategic initiatives. The outsourcing of infrastructure management enables internal IT teams to shift their focus from maintenance tasks to innovative projects that drive business value [5]. The SaaS TCO advantage provides financial benefits while simultaneously driving organizational agility and innovation.

5.2. Business Model Innovation: From Perpetual Licenses to Recurring Revenue

The SaaS model transformed software monetization by adopting recurring revenue streams instead of perpetual onetime licenses. The main subscription method uses a per-user per-month pricing model, but vendors also offer tiered pricing, usage-based billing, and freemium pricing structures.

The transformation leads to continuous revenue streams, which investors find highly attractive [7]. The new pricing model creates a direct connection between vendor incentives and customer achievement of success. The perpetual licensing model motivated vendors to make their first sales because they received no ongoing benefits from maintaining customer value. The subscription-based revenue model requires customers to stay subscribed over the long term for vendors to generate revenue. The lack of perceived value from customers leads to cancellation, which directly affects vendor revenue streams.

SaaS vendors established Customer Success teams to minimize customer churn and boost customer lifetime value (LTV) through their proactive functions that handle user onboarding, value realization, and engagement activities. The proactive approach of Customer Success differs from traditional support because it works to help users achieve important results with the software. The SaaS pricing model directly led to this post-sale relationship transformation.

Table 2. Ustrative 5-Year Total Cost Of Ownership (TCO) Comparison

Cost Category	On-Premise (Year 1)	On-Premise (Annual, Y2-5)	SaaS (Year 1)	SaaS (Annual, Y2-5)
Upfront Software License	$200,000	$0	$0	$0
Server & Infra. Hardware	$150,000	$0 (Upgrades in Y4)	$0	$0
Implementation & Custom.	$100,000	$0	$20,000	$0
Annual Maintenance/Support	$40,000	$40,000	(Included)	(Included)
IT Staff Overhead	$120,000	$120,000	$10,000	$10,000
Annual Subscription Fee	$0	$0	$120,000	$120,000
Annual Total	**$610,000**	**$160,000**	**$150,000**	**$130,000**
Cumulative 5-Year TCO	**$1,250,000**		**$670,000**	

5.3. The Democratization of Software and Its Impact on Enterprise Applications

The SaaS model reduced entry barriers for enterprise-grade tools because it eliminated significant initial expenses [9]. The democratization process made software accessible to small and medium-sized businesses (SMBs), which previously belonged to large enterprises [13]. The CRM market serves as an excellent illustration of this trend. CRM solutions during the 1990s required expensive on-premises installations of Siebel to operate. Salesforce.com introduced its SaaS CRM solution in 1999, which provided enterprise-grade functionality at affordable monthly rates [14]. The new pricing model allowed all business sizes to acquire and analyze customer data, which enhanced their marketing, sales, and service operations.

The SaaS model revolutionized the process through which organizations acquire software solutions. Enterprise software purchases used to fall exclusively under the control of CIOs and IT departments. The SaaS model introduced low-cost subscription options and browser-based signups, which allowed departments to acquire tools independently without following traditional IT procedures. The "Shadow IT" phenomenon brought both governance and security risks, yet it sped up innovation because teams could rapidly deploy customized solutions that met their specific needs.

The CIO transitioned from being an exclusive gatekeeper to becoming a strategic facilitator who oversees the management of various SaaS tools throughout the organization. SaaS technology gave departments the freedom to operate independently while forcing organizations to develop new, flexible approaches to technology strategy through collaborative governance models.

6. Challenges and the Road Ahead

6.1. Persistent Challenges in the SaaS Model: Security, Integration, and Vendor Lock-In

The numerous benefits of SaaS do not eliminate the substantial challenges that have intensified because of its broad adoption. Security and data privacy issues stand as the most vital concerns among all others. Organizations lose control of their core applications and sensitive data when they outsource them to third-party vendors, so they must depend on the provider’s security measures. The combination of data breaches from external attacks, insider threats, and misconfigured systems creates substantial risks in multi-tenant architectures because improper isolation could result in data leakage between customers [7]. The numerous configuration choices in complex SaaS applications lead to misconfigurations, which have become the primary reason for data exposure. The practice of vendor lock-in has become a significant issue that organizations face. The SaaS model, which was initially viewed as a flexible software licensing alternative, has evolved into a system that imposes substantial costs on organizations when they attempt to switch vendors. The combination of proprietary data formats, custom-built integrations, and user dependencies on specific workflows produces high levels of “stickiness,” which restricts portability [15]. Customers who use SaaS solutions become trapped within vendor ecosystems because they are exposed to price changes and technological stagnation.

The SaaS model originally promised IT simplicity, yet it has resulted in SaaS sprawl as an ironic consequence. Multiple SaaS applications are used by enterprises in large numbers, which results in a fragmented system of disconnected platforms. IT teams now manage distributed data while handling complex integration flows and inconsistent access control across vendors instead of maintaining one centralized on-premise platform. The widespread use of SaaS applications has driven the development of Cloud Access Security Brokers (CASBs) and SaaS Management Platforms (SMPs), which aim to unify control over decentralized systems. The SaaS model reduces specific infrastructure requirements yet creates a new level of complexity at the ecosystem level.

6.2. Market Trajectory and Future Trends: Verticalization, PaaS, and AI Integration

Gartner predicted in late 2018 that SaaS revenue would expand by 17.8% to achieve $85.1 billion in 2019 while maintaining its position as the largest segment of the public cloud market [16]. Market development has created multiple trends that will determine its future direction.

Vertical SaaS solutions represent a significant market trend because they provide industry-specific solutions for healthcare, manufacturing, and financial services sectors. Vertical SaaS products deliver advanced domain-specific features that meet the regulatory and operational needs of specific industries beyond the capabilities of horizontal business applications (e.g., CRM, HR) [17]. The specialized approach enables providers to deliver superior value to customers while building stronger customer loyalty.

The lines between SaaS and PaaS are becoming increasingly difficult to distinguish. Major vendors now provide platform access through APIs while offering development tools that let customers and developers extend core applications. The evolution of SaaS products enables them to operate as platforms instead of standalone tools by providing greater customization and extensibility.

The adoption of AI technology creates a significant technological transformation in SaaS. Applications now use AI and ML to automate workflows while providing predictive insights and improving user experiences. The competitive advantage of intelligence emerges through AI-driven chatbots in customer support platforms and ML-based demand forecasting in ERP systems.

The adoption of Product-Led Growth (PLG) strategies has become more prevalent among SaaS companies. The product itself serves as the primary driver of user acquisition through freemium or free trial models under the PLG approach, which replaces traditional sales-led methods. Users can directly experience value through this approach before making a purchase, which builds trust and minimizes purchase obstacles [18].

The evolution of SaaS extends past its traditional application delivery function. The digital enterprise operates through SaaS as its configurable, intelligent operating system, which abstracts both infrastructure and core business processes.

7. Conclusion

The transformation of software delivery methods from mainframe systems to SaaS systems marks a fundamental change in software development, distribution, and usage patterns. The early software systems operated under physical media and on-premises deployments through Waterfall methodologies, which failed to adapt to software’s flexible nature. The internet served as a catalyst that eliminated physical barriers to enable continuous delivery models. The unsuccessful attempt of Application Service Providers (ASPs) demonstrated the requirement for a mature ecosystem, which eventually materialized through the unification of cloud computing with multi-tenant architecture and Agile development and DevOps practices. The convergence of these elements made SaaS possible as a scalable, user-centric, cost-effective delivery model.

SaaS transformed software economics during 2020 by shifting from capital expenses to operational costs, which enabled businesses to adopt new strategies based on recurring revenue and Customer Success. Technology has made advanced tools accessible to everyone while creating equal opportunities for small businesses and transforming how organizations handle procurement and IT management. The future of SaaS development will bring vertical specialization, platform extensibility, and embedded artificial intelligence, which will transform SaaS from a delivery model into the core operating system of digital enterprises.

References

[1] “What is SaaS (Software as a Service)?” *Salesforce*. https://www.salesforce.com/saas/
[2] C. Tozzi, “Mainframe History: How Mainframe Computers Have Changed Over the Years,” *Precisely*. https://www.precisely.com/blog/mainframe/mainframe-history

[3] "Evolution of software development methodologies," *WADIC*. https://wadic.net/software-development-methodologies-evolution/

[4] Cardstack Team, "Software through the ages - Cardstack - medium," *Medium*, Jun. 30, 2020. [Online]. Available: https://medium.com/cardstack/software-through-the-ages-7ae7b3debfd7

[5] Oracle, "What is SAAS." https://www.oracle.com/applications/what-is-saas/

[6] S. J. Vaughan-Nichols, "The Pre-History of Software as a Service," Jan. 07, 2014. https://smartbear.com/blog/the-pre-history-of-software-as-a-service/

[7] Wikipedia contributors, "Software as a service," *Wikipedia*. https://en.wikipedia.org/wiki/Software_as_a_service

[8] W.-T. Tsai and P. Zhong, "Multi-tenancy and Sub-tenancy Architecture in Software-as-a-Service (SaaS)," in *IEEE Xplore*, Oxford, United Kingdom of Great Britain and Northern Ireland. [2014 IEEE 8th International Symposium on Service Oriented System Engineering]. Available: https://doi.org/10.1109/sose.2014.20

[9] V. Fryer, "The history of SAAS and the Revolution of Businesses | BigCommerce," *The BigCommerce Blog*. https://www.bigcommerce.com/blog/history-of-saas/

[10] S. MacDonald, "What is SaaS? A Data-Driven Guide to Software-as-a-Service," Jan. 01, 2025. https://www.superoffice.com/blog/saas/

[11] R. Madden, "The evolution of software delivery," *UXDX*. https://uxdx.com/blog/the-evolution-of-software-delivery/

[12] M. Evans, "SaaS and the Cloud vs. Custom-Built On-Premise Apps: What Finance Teams Need to Know," *Finario*. https://www.finario.com/saas-vs-on-premise-apps/

[13] "How fast-growing companies use SAAS to increase growth," *SuperOffice*. https://www.superoffice.co.uk/resources/articles/benefits-of-saas/

[14] J. White, "The SaaS takeover of the software industry," *Notion*, Oct. 30, 2020. https://www.notion.vc/resources/the-saas-takeover-of-the-software-industry

[15] M. Raza, "SaaS vs. PaaS vs. IaaS: What's the Difference and How to Choose," *BMC Blogs*. https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/

[16] N. Gagliordi, "Gartner predicts SaaS revenues to reach $85 billion in 2019," *ZDNET*, Sep. 12, 2018. [Online]. Available: https://www.zdnet.com/article/gartner-predicts-saas-revenues-to-reach-85-billion-in-2019/

[17] S. Schwartz, "2019 trends: Cocktail of SaaS applications becomes the norm," *CIO Dive*, Jan. 03, 2019. [Online]. Available: https://www.ciodive.com/news/2019-trends-cocktail-of-saas-applications-becomes-the-norm/544690/

[18] S. Patel, "SAAS trends: What's coming up ahead in 2020 – Sujan Patel," Jan. 15, 2020. https://sujanpatel.com/saas/saas-trends/