THE ECONOMICS OF AI-ASSISTED SOFTWARE DEVELOPMENT: FROM LOW-CODE TO VIBE CODING AND THE SAAS BUSINESS MODEL

Sándor SIMON

GÁL FERENC UNIVERSITY, FACULTY OF ECONOMICS
simon.sandor@gfe.hu

Abstract:

The rapid proliferation of artificial intelligence tools has fundamentally transformed the economics of software creation. This paper examines three interconnected transformations reshaping the software industry from the perspective of economics and business studies: (1) the unique economic characteristics of software as an information good — near-zero marginal cost, network effects, and high switching costs; (2) the democratisation of software development through the emergence of AI-assisted, natural-language programming paradigms, collectively termed 'vibe coding'; and (3) the principal risks and limitations arising from these new development paradigms, including code security vulnerabilities and structural platform dependency.

Keywords*: SaaS, vibe coding, low-code, AI-assisted development*

INTRODUCTION

Software has become the foundational infrastructure of the modern economy. From logistics and finance to healthcare and agriculture, virtually every sector now depends on software systems that collect, process, and communicate information. Yet for most of the industry's history, creating software required deep technical expertise — a significant barrier that effectively excluded business professionals, economists, and domain experts from participating directly in software production. That barrier is now eroding at an unprecedented pace.

Two converging forces are driving this change. First, the Software-as-a-Service (SaaS) model has reshaped the economics of software distribution, allowing even small teams to generate sustainable recurring revenues from narrowly scoped applications. Second, the emergence of large language model (LLM) based coding assistants — and their more radical offshoot, AI-native development platforms — has dramatically reduced the technical skill required to produce functional software. Together, these forces are opening new entrepreneurial pathways that are particularly relevant for students and graduates of economics, management, and business administration.

Building on prior research into low-code development environments, the paper argues that the barrier separating business domain experts from software production is dissolving at an accelerating pace, driven by a continuous progression from graphical low-code tools to AI-native, natural-language-driven development platforms. While this creates significant economic opportunities — particularly within the micro-SaaS segment — it also introduces risks that require careful analysis. The paper is intended primarily for students of economics and business administration, offering an accessible yet analytically rigorous entry point into one of the most consequential ongoing transformations of the digital economy. [11]

This paper builds directly on an earlier study by the present author , which examined the application of Microsoft PowerApps — a representative low-code development platform — to the automation of a sales process within a Microsoft Dynamics 365 Business Central environment. That study documented, among other findings, that a business-domain oriented developer can create production-quality enterprise applications with substantially lower technical overhead when using a low-code approach compared to conventional AL (Application Language) programming.

The present paper extends that analytical framework to the more recent and more radical shift represented by AI-assisted, natural-language-driven software development, situating both transitions within the broader economics of the software industry. [11]

The paper is structured around three pillars. Section 2 examines the economics of software as a distinctive class of information goods. Section 3 analyses how AI-assisted development — especially the 'vibe coding' paradigm — is transforming the production side of the software industry. Section 4 addresses the risks and limitations of these new paradigms, and Section 5 concludes with a synthesis and outlook.

THE ECONOMICS OF SOFTWARE AS AN INFORMATION GOOD

Software belongs to the category of information goods, a class of products whose economic characteristics differ fundamentally from those of physical commodities. The seminal analysis by Shapiro and Varian [10] identified three defining features of information goods that explain why software markets tend toward concentration, generate exceptional profit margins, and reward first-movers disproportionately: near-zero marginal cost, network effects, and high switching costs.

NEAR-ZERO MARGINAL COST

Unlike a physical product, software is reproduced at essentially zero marginal cost. Once the initial development investment — the fixed cost — has been incurred, each additional copy or user adds negligible cost to the producer. This property creates the potential for extreme economies of scale and gives information goods a distinctive cost structure: total costs are dominated by fixed costs (development, maintenance, infrastructure), while variable costs per unit approach zero . In the SaaS model, the practical manifestation of this property is the gross margin: mature SaaS businesses routinely achieve gross margins between 70% and 85%, far exceeding those achievable in manufacturing or most service industries. [4] Investors and acquirers recognise this characteristic, which partially explains why SaaS companies have historically been valued at revenue multiples of 8× to 15× — and in high-growth periods considerably higher — in contrast to the 1×–3× multiples typical in traditional industries. [5][10]

NETWORK EFFECTS AND LOCK-IN

A second distinguishing property of software is the network effect: the value of a product or platform increases as more users adopt it. This creates a self-reinforcing dynamic that tends to produce winner-take-most or winner-take-all outcomes in software markets . Platform businesses such as Salesforce, Microsoft 365, and ServiceNow derive a substantial portion of their competitive advantage from network effects: each new enterprise customer makes the platform more attractive to partners, developers, and system integrators, further entrenching the incumbent's position. Network effects compound with switching costs — the productivity losses, retraining investments, and integration disruptions associated with migrating to an alternative platform — to create durable competitive moats that are extremely difficult to erode. As Buday [1] argues, scalability and network effects together account for the majority of premium valuation multiples observed in the SaaS segment.[10]

THE SAAS MODEL: RECURRING REVENUE AND THE SUBSCRIPTION ECONOMY

The Software-as-a-Service delivery model — software delivered over the internet on a subscription basis rather than sold as a perpetual licence — has become the dominant commercial paradigm for enterprise and consumer software over the past two decades. By 2026, the global SaaS market is estimated at approximately USD 370–400 billion, and is projected to exceed USD 1 trillion by 2032, implying a compound annual growth rate of roughly 18% . More telling than

the absolute size is the penetration rate: by 2025, an estimated 99% of companies used at least one SaaS application, and the average enterprise operated over 130 distinct SaaS products .[13][8]

From an economic perspective, the subscription model transforms a volatile, lumpy revenue stream — dependent on irregular licence renewals — into a predictable annual recurring revenue (ARR) stream. This predictability significantly reduces the cost of capital for software businesses, since investors and lenders can more accurately forecast future cash flows. The resulting valuation premium over perpetual-licence peers has been documented empirically across multiple market cycles . At the same time, the subscription model imposes a continuous performance obligation on the vendor: unlike a one-time transaction, each subscription period represents a renewed customer decision, making customer retention — typically measured as net revenue retention (NRR) — the central operational metric of a SaaS business. [2]

THE AI REVOLUTION IN SOFTWARE DEVELOPMENT: FROM LOW-CODE TO VIBE CODING

The history of software development is, in part, a history of successive abstractions that have lowered the technical barrier to entry. Assembly language abstracted machine code; high-level languages abstracted assembly; integrated development environments and object-oriented frameworks abstracted low-level memory management; and the low-code/no-code movement of the 2010s abstracted syntax itself, allowing business users to construct applications through graphical interfaces and pre-built components. The present decade represents a further qualitative leap: artificial intelligence tools now allow software to be specified in natural language and generated automatically, with the human developer acting as director rather than craftsperson.

FROM LOW-CODE TO AI-ASSISTED DEVELOPMENT: A CONTINUUM

Simon [11] documented the practical consequences of the low-code paradigm in a real-world enterprise context, showing that the development of a Microsoft PowerApps application for sales process management within a Dynamics 365 Business Central environment required substantially less specialised programming knowledge than the alternative conventional approach using AL code. That study established that the low-code paradigm does not eliminate the need for domain knowledge and process understanding — indeed, these become more important as technical barriers decline — but it does substantially reduce the dependence on syntactical programming expertise. The low-code/no-code market has grown accordingly: by 2026, the market is estimated at approximately USD 30 billion, and Gartner has projected that by 2026 at least 75% of new enterprise applications will be built using low-code or no-code tools . [7]

AI-assisted development, represented by tools such as GitHub Copilot, Amazon CodeWhisperer, Cursor, and Windsurf, extends this trajectory by embedding large language model capabilities directly into the development workflow. Rather than merely providing pre-built components, these tools generate arbitrary code in response to natural-language specifications, debug errors autonomously, and explain existing code in plain language. The developer shifts from writing code line-by-line to reviewing, directing, and integrating AI-generated code — a change comparable, in its implications for skill requirements and productivity, to the introduction of computer-aided design in engineering or statistical software in economics.

VIBE CODING: NATURAL LANGUAGE AS THE NEW PROGRAMMING INTERFACE

The most radical manifestation of this trend has been termed 'vibe coding' — a concept introduced by Andrej Karpathy, co-founder of OpenAI and former Director of AI at Tesla, in a widely cited February 2025 post. Karpathy described an emerging practice in which the developer communicates intent to an AI system entirely in natural language, accepts the generated code

without reading it in detail, and iterates based on the observed behaviour of the resulting application rather than on an understanding of the underlying code. The cultural resonance of the concept was evidenced by Collins Dictionary naming 'vibe coding' its Word of the Year for 2025 , a recognition that signals the concept's penetration beyond the specialist programming community into mainstream discourse.[3]

From an economic perspective, vibe coding represents a further compression of the fixed cost of software development. Whereas traditional software development of a viable product might require an investment of USD 50,000–200,000 and a team of specialists over six to twelve months, AI-assisted development platforms such as Bolt.new, Lovable, and Replit have enabled functional prototypes to be produced by a single non-programmer in hours or days . This compression does not eliminate the economic value of professional software engineering for complex, safety-critical, or large-scale systems, but it substantially lowers the entry threshold for the large and growing category of narrowly scoped, domain-specific business applications.[12]

RISKS AND LIMITATIONS OF AI-ASSISTED SOFTWARE DEVELOPMENT

The democratisation of software development through AI tools carries significant risks that must be understood alongside the opportunities. This section examines three categories of risk: code quality and security, platform dependency, and economic concentration.

CODE QUALITY AND SECURITY

A consistent finding across multiple independent studies is that AI-generated code exhibits a significantly higher rate of security vulnerabilities than human-written code when the developer does not possess the expertise to evaluate the output. A 2025 arXiv study examining over 200,000 AI-generated code snippets found that 40% contained at least one exploitable security vulnerability, including SQL injection, authentication bypass, and data exposure risks . A parallel industry study reported that AI-generated code posed major security risks in nearly half of all development tasks. [9] These findings are particularly relevant for citizen developers and vibe coders who, by definition, may lack the security expertise to identify problems in generated code. [6]

The risk is compounded by the tendency of AI coding assistants to generate plausible-looking but functionally incorrect code — sometimes referred to as 'hallucination' in the LLM literature — particularly when the requested functionality is unusual or at the boundary of the model's training distribution. A developer who does not read the generated code critically may deploy applications with embedded logical errors that are difficult to detect until they cause operational failures.

PLATFORM DEPENDENCY AND THE "BUILD ON SAND" PROBLEM

A structural risk specific to the micro-SaaS and vibe coding ecosystem is the tendency to build on rapidly evolving or commercially uncertain platforms. As Simon [11] observed in the context of the Microsoft PowerApps/Dynamics 365 ecosystem, even within a mature enterprise vendor's platform, breaking API changes, licence restructuring, and feature deprecations can impose substantial migration costs on applications built on those platforms. This risk is substantially greater in the emerging AI development platform market, where products such as Bolt.new, Replit, and Lovable are early-stage companies whose commercial viability and API stability cannot be assured.

The practical implication for entrepreneurs and citizen developers is the importance of evaluating platform stability — the vendor's financial health, the maturity of their API versioning policy, the size and independence of the developer community — before committing to a platform as the foundation of a business application. Established platforms (Microsoft Power Platform, Salesforce, AWS Amplify) offer greater stability at the cost of higher entry costs and greater

vendor lock-in; newer, AI-native platforms offer greater creative flexibility and lower entry costs but carry higher existential risk.

CONCLUSION

This paper has examined the economics of software and the transformations currently reshaping the software industry through the lens of economics and business studies rather than computer science. Two principal conclusions emerge.

First, software exhibits distinctive economic properties — near-zero marginal cost, network effects, and high switching costs — that explain the exceptional profitability and valuation multiples of the SaaS segment and make an understanding of software economics a core competency for any business professional operating in a digitally intensive environment.

Second, the emergence of AI-assisted development tools — culminating in the 'vibe coding' paradigm recognised as Word of the Year 2025 — represents a qualitative discontinuity in the economics of software production. The fixed cost of creating a functional, narrowly scoped software product has declined by an order of magnitude. This is a direct continuation and amplification of the low-code trend documented by Simon [11]: just as low-code platforms such as Microsoft PowerApps enabled business analysts to build enterprise applications without deep programming knowledge, AI-native development platforms extend this capability to the point where the primary limiting factor is domain knowledge rather than technical skill. The risks associated with this democratisation — code security vulnerabilities and structural platform dependency — are real but manageable: they call for sound economic judgment and technical literacy rather than deep programming expertise, competencies that business and economics graduates are well positioned to develop.

REFERENCES

[1] Buday, M. (2025, June 26). Scalability vs. network effects in valuations. Clearly Acquired. Retrieved from https://www.clearlyacquired.com/blog/scalability-vs-network-effects-in-valuations

[2] Clfi (2025, November 4). The economics of SaaS explained: Recurring revenues and growth. Retrieved from https://clfi.co.uk/resources/economics-of-saas-recurring-revenues-and-growth/

[3] Collins Dictionary (2025, November 5). Collins Word of the Year 2025: AI meets authenticity as society shifts. Retrieved from https://blog.collinsdictionary.com/language-lovers/collins-word-of-the-year-2025-ai-meets-authenticity-as-society-shifts/

[4] Fiscallion (2026, April 16). SaaS gross margin benchmark: What investors actually expect at every stage. Retrieved from https://www.fiscallion.io/blog/saas-gross-margin-benchmark-what-investors-actually-expect-at-every-stage

[5] Hectelion (2025, October 30). Software and SaaS valuation: Economic value and AI impact. Retrieved from https://www.hectelion.com/en/publications/software-and-saas-valuation-economic-value-and-ai-impact

[6] Koutcheme, C., Sarsa, S., Leinonen, J., & Hellas, A. (2025). Security vulnerabilities in AI-generated code. arXiv preprint arXiv:2510.26103. Retrieved from https://arxiv.org/abs/2510.26103

[7] Netclues (2025, April 15). Low code no code platforms 2025. Retrieved from https://www.netclues.com/blog/low-code-no-code-platforms-2025

[8] Precedence Research (2025). Low-code and no-code development platforms market size 2025–2034. Retrieved from https://www.precedenceresearch.com/low-code-and-no-code-development-platforms-market

[9] Security Today (2025, August 5). AI-generated code poses major security risks in nearly half of all development tasks. Retrieved from https://securitytoday.com/articles/2025/08/05/ai-generated-code-poses-major-security-risks-in-nearly-half-of-all-development-tasks

[10] Shapiro, C., & Varian, H. R. (1999). Information rules: A strategic guide to the network economy. Harvard Business School Press.
[11] Simon, S. (2023). PowerApps Development to a Microsoft Dynamics 365 Business Central Based Sales Process management In : Gál Ferenc University Deliberationes Scientific Journal Vol. 15. Spec. Ed. No. 1/ 2022, https://doi.org/10.54230/Delib.2022.K.SZ.89
[12] Tizbi (2026, April 5). AI development in 2026: Vibe coding guide. Retrieved from https://tizbi.com/articles/vibe-coding-business-guide-2026
[13] Zylo (2026, February 12). 175+ unmissable SaaS statistics for 2026. Retrieved from https://zylo.com/blog/saas-statistics/