Journal of Software Engineering and Applications, 2026, 19(1), 1-6
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

The New Software Cost Curve under Agentic AI: Development Economics, Pricing, and Labor Impacts

Wasim Haque

Independent Researcher, Woodstock, GA, USA
Email: haque.wasim@gmail.com

How to cite this paper: Haque, W. (2026) The New Software Cost Curve under Agentic AI: Development Economics, Pricing, and Labor Impacts. *Journal of Software Engineering and Applications*, **19**, 1-6.
https://doi.org/10.4236/jsea.2026.191001

Received: January 1, 2026
Accepted: January 25, 2026
Published: January 28, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

Agentic AI—software agents that plan, generate, test, and iterate across the software lifecycle—is bending the software cost curve. We merge two complementary analyses: 1) a development-focused comparison between a traditional human-centric Agile team and an agentic workflow, and 2) an extension from build cost to total cost to deliver (build, run, sell, and risk), including implications for SaaS and non-SaaS pricing and the software labor market. Using an illustrative benchmark, we show how build costs can collapse (irrespective of methodology you use, old or new ones like T3D [1]) by more than an order of magnitude while time-to-market compresses from quarters to weeks. The binding constraints then shift toward operations, distribution, and risk management. We propose a pricing pressure matrix and a practical playbook for packaging, governance, and workforce reskilling over a 1 - 7-year horizon.

Keywords

Agentic AI, Software Economics, Total Cost to Deliver, SaaS Pricing, Software Labor Market, AI Risk Governance, DevOps, SRE, T3D, Test and Defect Driven Development

1. Introduction

Software engineering economics has historically been shaped by scarce human labor, long feedback loops, and coordination overhead. Agentic AI introduces a new production function: goal-conditioned agents can interpret intent, generate and refactor code, produce tests, and iterate through tooling. The economic question is not whether productivity rises—it does—but how costs shift from build to

run, sell, and risk as software becomes easier to produce [2] [3].

This paper merges two complementary analyses into one cohesive framework. First, we compare a traditional human-centric Agile delivery model to an agentic workflow using an illustrative benchmark. Second, we extend from development cost to total cost to deliver (TCD), pricing power, and labor-market implications across SaaS and non-SaaS software. Our results are directional; organizations should calibrate to their domain, regulatory environment, and operational maturity.

2. Conceptual Framework

2.1. Agentic AI-Driven Software Development

We define agentic AI-driven development as workflows in which AI systems act as autonomous or semi-autonomous agents—planning, generating, testing, and integrating changes—while humans provide intent, constraints, review, and risk governance. Compared with assistive tooling, agentic systems shift more work into execution by delegating to tool-using agents.

2.2. Total Cost to Deliver (TCD)

Development expense is only one component of software economics. We model total cost to deliver as: TCD = Build + Run + Sell + Risk. When build becomes cheap, run (operations and compute), sell (distribution and retention), and risk (security, compliance, reliability) dominate long-run economics [4]-[6].

3. Baseline: Traditional Agile Delivery

3.1. Team Structure and Cost Model

A representative small-to-mid product team often includes backend, frontend, DevOps, QA, product management, and design. This model is effective but labor-intensive, and coordination and rework can materially affect cost and schedule [1].

- 2 Senior Backend Engineers—$300,000 combined.
- 1 Frontend Engineer—$130,000.
- 1 DevOps Engineer—$140,000.
- 1 QA Engineer—$90,000.
- 0.5 Product Manager—$60,000.
- UI/UX Designer—$100,000.
- Enterprise Tooling—$180,000.

Total Estimated Annual Cost: $1.0 M/yr.

3.2. Delivery Timeline and Coordination Overhead

Traditional delivery frequently spans multiple quarters—discovery, MVP buildout, expansion, hardening, and launch—especially in environments with multiple stakeholders and integration dependencies. Empirical DevOps research suggests

that organizational practices and feedback loops are strong drivers of delivery performance and outcomes [3].

4. Agentic Workflow Effects: Build Cost and Time-to-Market

4.1. Build Cost Collapse (Illustrative Benchmark)

In an illustrative benchmark, first-year build investment declines from approximately $1.0 M in a team-based model to under $300,000 in direct agentic workflow costs. The purpose of this estimate is to highlight the magnitude and direction of change as shown in in Figure 1, not to predict a universal ratio.

- 0.5 Full-stack Engineer—$100,000.
- 0.5 DevOps Engineer—$70,000.
- 0.5 Product Manager—$60,000.
- 0.5 UI/UX Designer—$50,000.
- Enterprise Tooling (reduced licensing by 30%)—$120,000.

Total Estimated Annual Cost: $400,000/yr.

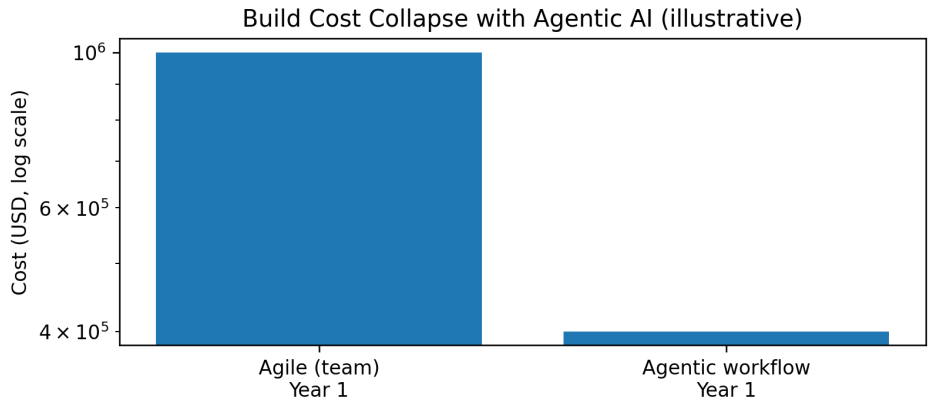

Figure 1. Build cost collapse with agentic AI (log scale; illustrative).

4.2. Cycle-Time Compression and Market Crowding

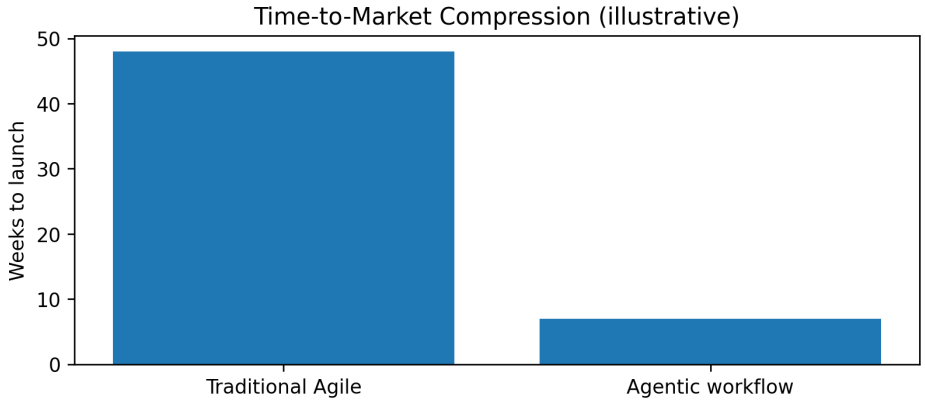

Figure 2. Time-to-market compression (illustrative).

Time-to-market compresses from roughly 48 weeks to approximately 6 - 8 weeks

as indicated in Figure 2. Faster iteration accelerates experimentation but also increases market crowding and the speed of feature parity. Organizations should expect faster competitive cycles and greater pressure to differentiate beyond feature sets.

5. Where the Cost Moves: Run, Sell, and Risk

5.1. Run-Cost Gravity (Operations and Inference)

As more product behavior is mediated by models, inference and orchestration can become material operating costs. Even if unit inference prices fall, total spend can rise with usage. Reliability and observability become more important as the operational surface area grows [4].

5.2. Sell-Cost Dominance under Feature Commoditization

When competitors can reproduce features quickly, distribution, integrations, partnerships, and retention become decisive. In many categories, customer acquisition and customer success can dominate long-run economics, making cost-of-growth a key strategic variable.

5.3. Risk Costs and Governance

Risk costs include breaches, compliance failures, outages, and reputational damage. Agentic speed without governance can increase expected loss (Figure 3). Practical controls include least-privilege tool permissions, staged rollouts with canaries, continuous evaluation, and auditability [5] [6].

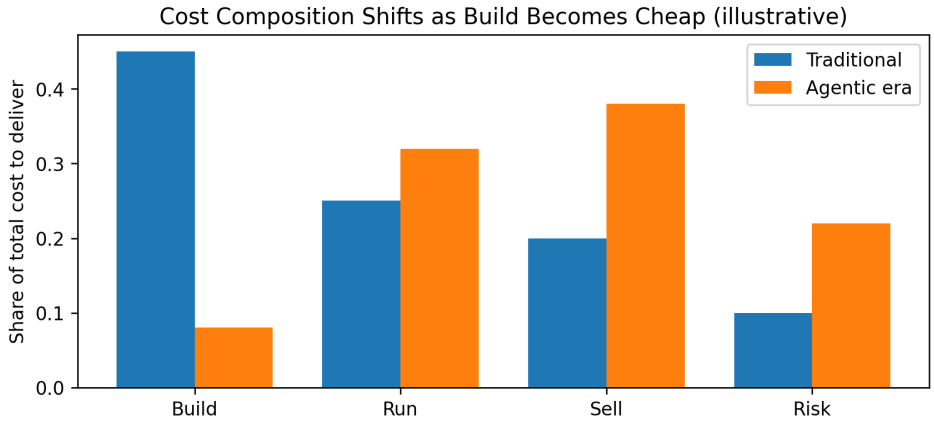

Figure 3. Cost composition shifts as build becomes cheap (illustrative shares).

6. Pricing, Packaging, and Profit

6.1. Pricing Pressure Matrix

We propose a conceptual pricing matrix (Figure 4): products whose value is primarily feature-based and easy to replicate face compression, while regulated, trusted, or outcome-linked products can sustain or expand pricing power. Switching costs and integration depth further mediate competitive dynamics.

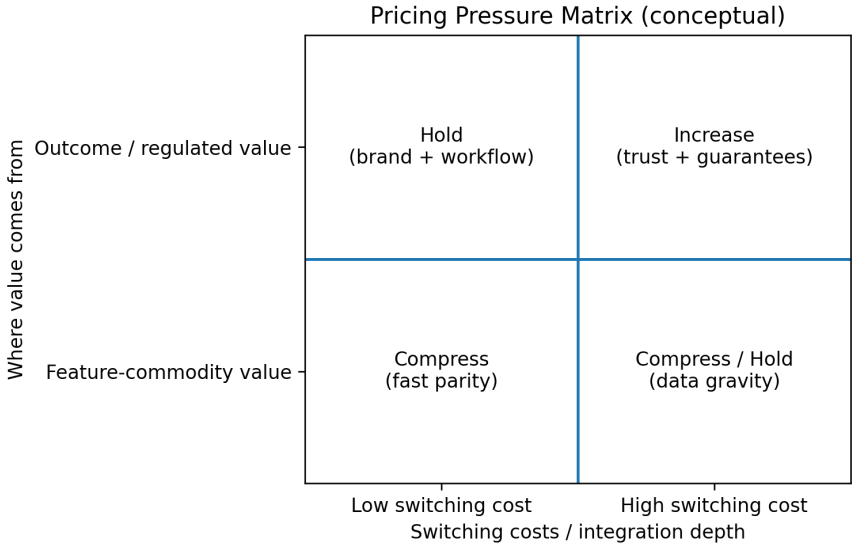

Figure 4. Pricing pressure matrix: where prices compress vs hold vs increase (conceptual).

6.2. Packaging Playbook for SaaS

SaaS packaging will likely bifurcate into seat-based, usage-based, outcome-based, and platform + ecosystem models. In commoditized categories, prices compress; in high-stakes categories, buyers pay for guarantees, auditability, and measurable outcomes.

6.3. Non-SaaS Software: Services and Internal Portfolios

Services firms may see time-and-materials defensibility erode; buyers compare vendors on governance, integration outcomes, and risk management. Enterprises may build more internal software as marginal build cost falls, but without portfolio governance this can create sprawl and operational burden.

7. Labor Market and Skills

7.1. Role Shifts and Skill Rebalancing

Agentic AI changes the composition of software work. Routine implementation becomes less scarce, while architecture, platform engineering, security, reliability, data engineering, and product judgment grow in relative value [3] [4].

7.2. Outlook (1 - 7 Years)

In a 1-2-year horizon, build-cost declines and cycle-time compression increase product supply and intensify competition. Over 3 - 4 years, trust, compliance, and distribution become primary differentiators. Over 5 - 7 years, value capture shifts further toward outcomes, ecosystems, and regulated assurance [7].

8. Recommendations

8.1. SaaS Companies

Re-price around outcomes and risk reduction rather than feature counts; invest

in trust as a product via auditability, data policies, SLAs, and incident readiness; and build distribution moats through integrations and ecosystems.

8.2. Services Firms

Shift from hours billed to packaged delivery and measurable outcomes; productize reusable components; and offer AI SDLC governance and compliance enablement as premium services.

8.3. Enterprises

Stand up an AI SDLC with review standards, scanning, audit logs, and clear ownership; build internal platforms with templates and policy-as-code; and measure portfolio ROI to retire low-value apps.

9. Conclusion

Agentic AI bends the software cost curve by collapsing build cost while increasing the economic importance of trust, distribution, and outcomes. Organizations that treat governance and reliability as product capabilities can expand margins even underpricing pressure; feature-only competitors will struggle.

Acknowledgements

I thank the software engineering community for advancing empirical delivery measurement, reliability, and security practices.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Haque, W. (2025) Test and Defect Driven Development (T3D): A Novel Approach to Software Development. *Journal of Software Engineering and Applications*, **18**, 139-147. https://doi.org/10.4236/jsea.2025.184009

[2] Brooks, F.P. (1995) The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley.

[3] Forsgren, N., Humble, J. and Kim, G. (2018) Accelerate: The Science of Lean Software and DevOps. IT Revolution Press.

[4] Beyer, B., Jones, C., Petoff, J. and Murphy, N.R. (2016) Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media.

[5] National Institute of Standards and Technology (NIST) (2023) AI Risk Management Framework (AI RMF 1.0). NIST. https://www.nist.gov/itl/ai-risk-management-framework

[6] ISO/IEC (2022) ISO/IEC 27001:2022 Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements. International Organization for Standardization.

[7] U.S. Bureau of Labor Statistics (BLS) (2024) Occupational Outlook Handbook: Software Developers, Quality Assurance Analysts, and Testers. U.S. Department of Labor.