SOFTWARE COMMODITIZATION IN THE AI-FIRST ECONOMY: FROM IMPLEMENTATION MOATS TO CONTRIBUTION MARKETS

Marius-Constantin Dinu
Alpha Omega Labs, Austria
dinu.marius.constantin@gmail.com
Global Executive MBA candidate, WU Executive Academy (WU Vienna)
and Carlson School of Management, University of Minnesota

ABSTRACT

AI-assisted software generation changes the economics of software scarcity. We argue that value does not disappear when recognizable interfaces and workflows become easier to reproduce. It moves toward assets and relationships that remain controlled, trusted, licensed, institutionally embedded, or agent-accessible. We develop five directional hypotheses. (1) Implementation moats can weaken for visible and evaluable patterns. (2) Defensibility shifts relatively toward complementary assets. (3) Access subscriptions face pressure when generic functionality is not bundled with trust, rights, compliance, or integration depth. (4) Governed service layers create new monetization options for agents. (5) Traceable contribution markets may route value to source assets and action endpoints when attribution, provenance, rights metadata, and payment rails are sufficient. Rather than treating commoditization as only a coding-productivity problem, we develop a multidisciplinary systems account that links software-engineering evidence, resource-based and information-systems strategy, intellectual-property scholarship, provenance and attribution methods, data economics, and agent-commerce infrastructure around one question: what remains scarce when implementation becomes easier to reproduce? We triangulate the argument with a qualitative case study of Myflix and recent descriptive market evidence. The contribution is integrative rather than a claim of universal proof: we connect existing theory to AI-assisted software reproduction and propose a bounded mechanism for moving from access monetization toward trusted capability, governed agent actions, and attributable contribution to generated outputs.

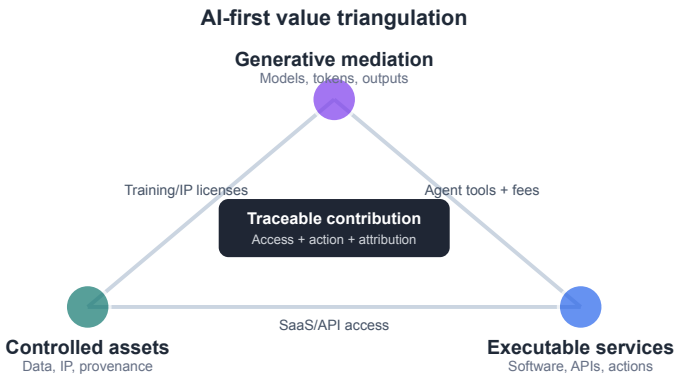


Figure 1: AI-first value triangulation. The figure defines three value domains used throughout the paper: controlled assets such as data, intellectual property, rights, and provenance; executable services such as software, application programming interfaces, and governed action endpoints; and generative mediation through models, tokens, prompts, and outputs. The center marks traceable contribution, where access, action, and attribution can be governed together.

1 INTRODUCTION

Software has historically combined two forms of scarcity. The first is implementation scarcity: the knowledge, time, and engineering coordination required to build a working system. The second is complementary scarcity: data, rights,

distribution, trust, compliance, brand, integration depth, and operational reliability. Generative AI changes the relative importance of these two forms. It does not remove the need for engineering judgment, testing, product taste, legal clearance, or operational governance. However, it plausibly reduces the cost of reproducing recognizable implementation patterns, especially when the target behavior can be described through examples, screenshots, tests, and iterative natural-language feedback.

Commoditization of software denotes a decline in the strategic scarcity of implementation for common categories of functionality. The term is narrower than a claim that software has no value. A dashboard, media browser, scheduler, internal workflow tool, CRM screen, or API wrapper may still require careful engineering, but the visible pattern can often be described, regenerated, repaired, and extended by AI-assisted systems. If implementation cost falls, then competitive advantage moves to whatever remains hard to imitate, hard to access, or hard to substitute.

The research question is:

> How does AI-assisted software generation change the sources of competitive advantage, IP protection, and monetization when software interfaces, workflows, and implementation patterns become inexpensive to reproduce?

Figure 1 summarizes our multidisciplinary framing. We treat AI-first value capture as the interaction of three scarcities. *Controlled assets* include data, IP, provenance, rights, trust, and legally governed access. *Executable services* include software, APIs, tools, workflows, and agent-action endpoints. *Generative mediation* includes models, prompts, tokens, generated outputs, and third-party context. Current business models often monetize one vertex or one edge: data deals and licensing monetize controlled assets; SaaS and API pricing monetize executable services; token billing monetizes model-mediated output; agentic commerce begins to monetize the service–model edge. The most defensible position we theorize is the governed intersection, where access, action, and attribution are priced together.

Recent market data make the question more precise rather than more deterministic. Eurostat reports that 19.95% of EU enterprises with at least 10 employees used AI technologies in 2025, while the OECD reports 20.2% firm AI adoption across OECD countries where data are available (Eurostat, 2025; OECD, 2026). The U.S. Census Bureau's 2026 BTOS AI supplement estimates 18% firm-level AI use in at least one business function during November 2025–January 2026, rising to 32% on an employment-weighted basis (Bonney et al., 2026). These numbers indicate rapid diffusion, not saturation. They also show strong size and sector gradients, which supports our thesis that complementary capability and organizational resources matter. At the same time, Stack Overflow's 2025 developer survey reports that 84% of developers use or plan to use AI tools, but 46% do not trust AI-output accuracy and 45% report time-consuming debugging of AI-generated code (Stack Overflow, 2025). We therefore treat AI-first commoditization as an uneven pressure on implementation scarcity, not as a completed market state.

The question matters for both software strategy and information-systems research. In strategy terms, implementation has often been treated as a resource, a capability, or a barrier to entry. In information systems, debates about whether IT matters have long asked whether standardized technology can confer durable advantage. In legal and platform terms, the rise of generative AI adds copyright, data provenance, interface interoperability, attribution, and agent-mediated transaction questions. In product terms, firms face a practical problem: if a recognizable software workflow can be reproduced quickly, then charging only for access to the workflow may become harder. The defensible revenue base may instead depend on data, rights, trust, compliance, distribution, transaction control, machine-readable services, and traceable contribution markets that compensate source assets when they measurably shape generated outputs or actions.

We organize the analysis around five tensions.

First, AI coding productivity evidence is mixed. Peng et al. (2023) report faster task completion in a controlled GitHub Copilot study, while METR (2025) report that experienced open-source developers in a 2025 field RCT were slower with AI assistance in the measured setting. Repository-level benchmarks and systems such as SWE-bench, SWE-agent, and Agentless show growing automation capability, but also clarify that success depends on localization, evaluation, tool design, and task structure (Jimenez et al., 2024; Yang et al., 2024; Xia et al., 2024). We therefore make a conditional, not universal, claim: AI can reduce reproduction cost for some implementation patterns, especially when scoped and iterated, but productivity is not guaranteed.

Second, implementation is only one component of competitive advantage. A resource decomposition separates implementation, data, rights, trust, network position, compliance, and service access. AI capability lowers the relative weight and cost of implementation while increasing the strategic importance of assets that remain scarce. The monetization pressure condition follows, and it is a comparison of two sums rather than of two terms: subscription defensibility weakens when the customer's total cost of leaving — the expected cost of obtaining a substitute *plus* the switching,

trust, compliance, and integration barriers — falls below the subscription price plus whatever value premium the incumbent genuinely carries. Cheaper replicas therefore bite only where the remaining barriers are thin relative to the price being charged; Section 3.3 states the condition formally.

Third, we use the Myflix case to make the mechanism observable. A user supplied a recognizable software category, feedback, screenshots, and feature requests. An AI coding agent helped produce a multi-workspace TypeScript application with web, API, worker, domain, media, provider, and UI packages. The resulting system includes browsing, playback, offline retention, provider boundaries, account/profile state, settings, downloads, subtitles, and TV Mode features. The generic browsing shell is imitable; the controlled provider rules, local media provenance, profile state, validation routines, playback/transcoding behavior, and source configuration boundaries are more defensible.

Fourth, copyright, API law, data provenance, and service-layer monetization bound the commoditization thesis. Supreme Court of the United States (2021) shows that software API declarations and interoperability are legally complex; Lemley (2024) highlights how generative AI strains copyright doctrine; and data-provenance research shows that consented and controlled data resources can become strategically scarce (Longpre et al., 2024). At the same time, primary documentation for the Model Context Protocol, OpenAI Apps SDK, and Stripe agentic commerce suggests that the interface of software is expanding beyond human-facing screens into agent-readable tools, components, and payment flows (Model Context Protocol, 2026; OpenAI, 2026a; Stripe, 2026b). These service layers create new monetization surfaces when the visible UI is easier to reproduce.

Fifth, traceable contribution markets offer a constructive path beyond defensive IP alone. Training-data attribution, provenance standards, data valuation, and machine-readable payment protocols are not mature enough to support exact token-origin royalties. They are mature enough to motivate a bounded mechanism: source assets can publish rights and provenance metadata, generated-output or agent-action events can record context and revenue, attribution systems can assign confidence-weighted contribution shares, and payment rails can settle micro-royalties or service-action fees (Koh & Liang, 2017; Pruthi et al., 2020; Park et al., 2023; Jiao et al., 2025; Ghorbani & Zou, 2019; World Wide Web Consortium, 2013; Coalition for Content Provenance and Authenticity, 2025; Coinbase Developer Platform, 2026; Linux Foundation, 2026).

2 Related Work

2.1 AI Productivity and Task Exposure

The broader generative-AI productivity literature establishes both the promise and the heterogeneity of AI assistance. Noy & Zhang (2023) report that ChatGPT improved speed and quality in an experimental writing-task setting. Brynjolfsson et al. (2025) study a deployed customer-support assistant and find average productivity gains with especially large gains for less-experienced workers. Eloundou et al. (2023) frame large language models as general-purpose technologies and estimate substantial task exposure when LLMs are combined with software tooling. Brynjolfsson et al. (2017) caution that general-purpose technologies often require complementary innovation, organizational redesign, and measurement lags before their productivity effects appear in aggregate statistics.

Skills-centered exposure research sharpens the same distinction. Chopra et al. (2025) introduce the Iceberg Index as a measure of the wage value of occupational skills that AI systems can technically perform. The index measures technical exposure, not displacement outcomes or adoption timelines. Its empirical contrast is useful for this paper: visible AI adoption in computing and technology accounts for 2.2% of wage value, while broader technical exposure across cognitive, administrative, financial, and professional work reaches 11.7%, or about $1.2 trillion. That gap supports our use of market statistics as boundary evidence rather than as a complete measure of software commoditization.

We distinguish software commoditization from programmer productivity. Productivity studies ask whether a person completes a measured task faster or better. Commoditization asks whether buyers and competitors can obtain an acceptable substitute more cheaply. The two can diverge. AI may help a non-expert assemble a substitute even when an expert maintainer faces overhead integrating AI into a complex repository. Foundation-model research reinforces this distinction: broad pretraining and adaptation create general-purpose capabilities, but those capabilities inherit data, evaluation, safety, and deployment risks (Bommasani et al., 2021).

2.2 AI-Assisted Software Engineering

AI-assisted programming began in the mainstream as completion and suggestion, but current research increasingly evaluates repository-level behavior. Peng et al. (2023) provide empirical evidence that GitHub Copilot can reduce completion time in a controlled programming task. The relevance to commoditization is direct: if a bounded feature

can be implemented faster, then implementation scarcity declines for that class of task. However, controlled task completion is not the same as building and operating a production system.

SWE-bench moves the question closer to real-world software maintenance. It evaluates whether language models can resolve GitHub issues by editing real repositories and passing tests (Jimenez et al., 2024). SWE-agent adds an agent-computer interface for navigating, editing, and testing software repositories (Yang et al., 2024). Agentless argues that some apparent agent capability can be achieved with simpler localization and repair pipelines (Xia et al., 2024). We use these works because commoditization does not require fully autonomous software companies. It requires enough reduction in reproduction cost that implementation alone becomes less rare.

Agentic and neuro-symbolic systems explain why the relevant interface is broader than code completion. Dinu et al. (2025) introduce SymbolicAI as a framework that treats LLMs as semantic parsers connecting generative models, solvers, probabilistic programming, and explainable computational graphs. Patel et al. (2024) study web agents on WebArena and report a 31% task-completion improvement after fine-tuning on synthetic self-improvement data. Dinu (2024) frames broad AI as a generalization problem across domains and argues that neuro-symbolic approaches can support adaptable computational graphs when gradient-based tuning is infeasible. These works support the agent-service and workflow-composition parts of the paper. We treat them as technical context, not as independent evidence that software markets have already commoditized.

The 2026 benchmark picture strengthens and qualifies that claim. Stanford's AI Index reports rapid progress on software and agentic benchmarks, including large one-year improvements on SWE-bench Verified and OSWorld (Stanford Institute for Human-Centered Artificial Intelligence, 2026). However, OpenAI's 2026 analysis argues that SWE-bench Verified is increasingly contaminated and no longer cleanly measures frontier coding capability; it reports progress slowing, from 74.9% to 80.9% while also finding material test-design or problem-description issues in 59.4% of an audited hard subset (OpenAI, 2026b). We therefore read repository benchmarks as evidence of capability pressure and evaluation saturation, not as proof that autonomous agents can replace all software engineering.

The code-generation literature also prevents a purely optimistic account. Codex and AlphaCode demonstrate that large models can synthesize non-trivial programs from natural-language or contest-style specifications (Chen et al., 2021; Li et al., 2022). Yet user studies and security analyses show that generated code still introduces workflow friction, expectation gaps, and security risks (Vaithilingam et al., 2022; Barke et al., 2023; Pearce et al., 2022; Perry et al., 2022). In commoditization terms, AI can lower the cost of an acceptable substitute without making the substitute operationally mature, secure, or strategically defensible.

The productivity evidence is not one-directional. METR (2025) report that experienced open-source developers were slower with early-2025 AI assistance in their randomized study, even though participants expected AI to help. We treat this result as a central caution: AI does not automatically improve engineering throughput. Friction can come from understanding generated code, maintaining repository-specific constraints, debugging false starts, or managing context. The AI effect is therefore conditional: reproduction cost falls most strongly where goals are visible, requirements are decomposable, feedback loops are short, and evaluation is available.

2.3 Strategy, Resources, and IT Commoditization

The resource-based view argues that sustained competitive advantage depends on resources that are valuable, rare, imperfectly imitable, and non-substitutable (Wernerfelt, 1984; Barney, 1991; Peteraf, 1993). Asset-stock accumulation and causal ambiguity make some resources difficult to buy or copy quickly (Dierickx & Cool, 1989). If AI reduces the cost of reproducing software implementation, then implementation loses some rarity and imperfect imitability. The RBV prediction is not that firms lose all advantage. It is that advantage migrates to assets that remain valuable and difficult to imitate.

The specific mechanism this paper reuses is older than the resource-based view's general form. Teece (1986) asks why innovators so often fail to capture the value of their own innovations, and answers that when imitation is easy, returns accrue to whoever controls the *complementary assets* an innovation must pass through to reach a customer. That is the argument we are restating: cheap AI-assisted reproduction is a shift in the appropriability regime for a broad class of software, and Teece's prediction for a weak appropriability regime is precisely that value moves to the complements. Our contribution is not the mechanism but its application and measurement in this setting.

Dynamic capabilities theory emphasizes the ability to integrate, build, and reconfigure competencies in changing environments (Teece et al., 1997). We apply that logic to an AI-first economy because the pace of imitation can increase. The firm that merely owns a software artifact may be less protected than the firm that can continuously adapt data pipelines, legal rights, distribution partnerships, compliance controls, pricing models, and machine-readable interfaces.

Information-systems research provides a bridge between IT as commodity infrastructure and IT as firm-specific capability. Mata et al. (1995) ask when IT can provide sustained advantage; Clemons & Row (1991) emphasize structural differences; and Powell & Dent-Micallef (1997) show that human, business, and technology resources interact. Bharadwaj (2000) finds that IT capability can be associated with firm performance, while Melville et al. (2004) synthesize IT business value as contingent on internal, external, and complementary resources. Wade & Hulland (2004) reviews how IS resources can matter when embedded in complementary resources and organizational context. Carr (2003) famously argues that IT's strategic importance declines as it becomes standardized and ubiquitous, while Porter (1996; 2001) argues that operational effectiveness and Internet technology do not abolish distinct strategic positioning. McAfee & Brynjolfsson (2008) add that IT can intensify competition because rivals can propagate process innovations rapidly. Digital-business strategy extends this view by treating digital resources, platform position, and cross-boundary recombination as central to business strategy (Bharadwaj et al., 2013). Recent generative-AI strategy work reaches a closely related conclusion at the model-market level: Azoulay et al. (2024) argue that competition in generative AI depends on appropriability and control over complementary assets such as infrastructure, data, distribution, and capabilities. These literatures converge on a reconciled view: generic implementation can commoditize, while organization-specific capability, rights, data, trust, and service access remain strategic.

2.4 Digital Innovation, Platforms, and Ecosystems

Digital innovation research explains why software is particularly prone to recombination. Yoo et al. (2010) describe layered modular architecture across devices, networks, services, and content. Nambisan et al. (2017) argue that digitization changes innovation boundaries, agency, and the relation between process and outcome. Henfridsson et al. (2018) distinguish design recombination from use recombination and shift attention toward digital resources as reusable building blocks. Zittrain (2006) shows how generative computing environments enable broad third-party creation while also creating security and governance pressure.

Platform and ecosystem research extends this argument from artifacts to governance. Tiwana et al. (2010) connect platform evolution to architecture, governance, and environmental dynamics. Gawer & Cusumano (2014) describe industry platforms as foundations for ecosystem innovation. Adner (2017) treats ecosystems as structured interdependence among actors, activities, positions, and links. Jacobides et al. (2018) emphasize modularity and complementarities as conditions under which ecosystems emerge. We draw on these works for the service-layer argument: when implementation is reproducible, the strategic question shifts toward who governs the complementary modules, access rules, and interdependencies.

Boundary-resource and platform-architecture research sharpens that point. Platforms depend on modular architectures and governed interfaces (Baldwin & Woodard, 2009). Platform envelopment shows that adjacent platform owners can bundle functionality and pressure standalone offerings (Eisenmann et al., 2011). Developer ecosystems invert parts of the firm by allowing external complementors to create value on controlled platforms (Parker et al., 2017). Boundary resources mediate the tension between external contribution and platform control (Ghazawneh & Henfridsson, 2013; Eaton et al., 2015). In an AI-first economy, agent-facing APIs, MCP servers, and commerce endpoints can be interpreted as new boundary resources: they permit external agents to act, but only through controlled service contracts.

2.5 Business Models, Monetization, and Lateral Industry Evidence

Business-model research provides the value-capture side of the commoditization question. Teece (2010) define business models around value creation, delivery, and capture; Chesbrough & Rosenbloom (2002) show that a business model can unlock economic value from technology but can also constrain subsequent search. Shapiro & Varian (1998) analyze information goods through pricing, lock-in, network effects, and rights management, all of which remain relevant when software interfaces become easier to reproduce.

Lateral industry evidence sharpens the analogy. Vroom et al. (2021) describe how digitization changed the music value chain and pushed the industry toward streaming, licensing, and rights-based models after file sharing reduced distribution scarcity. Christensen et al. (2002) connect disruptive growth to new markets and business models rather than sustaining improvements alone. Lakhani et al. (2015) present GE's Industrial Internet initiative as an example of moving from equipment and software artifacts toward data-enabled, outcome-oriented services. These cases do not prove the AI software thesis. They motivate a pattern: when a technical layer becomes cheaper or more replicable, durable value often shifts toward rights, business-model design, operational data, and outcome contracts.

2.6 IP, Data, Provenance, and Interoperability

Software commoditization is not only an engineering question. Reproduction can be technically possible but legally risky, ethically unacceptable, or commercially infeasible. Copyright law, trademark law, trade secrets, data rights, contractual access, and platform terms constrain what can be copied. Supreme Court of the United States (2021) is important because the Supreme Court's Java API decision treats API declarations and fair use in a fact-specific interoperability context. The decision does not imply that all API copying is lawful; rather, it illustrates that interface reuse and software competition are legally nuanced.

Generative AI adds further uncertainty. Lemley (2024) argues that generative AI stresses conventional copyright categories because training, outputs, style, and substitution do not map cleanly onto earlier doctrines. Data documentation research adds a governance foundation: Gebru et al. (2021) propose datasheets for datasets, and Bender & Friedman (2018) propose data statements to make data provenance, population, and use limits explicit. Longpre et al. (2024) audit dataset licensing and attribution at scale and document pervasive documentation gaps. As generic implementation becomes cheaper, consented data, licensed content, and auditable provenance can become more important sources of scarcity.

The adjacent AI-governance literature makes this operational rather than rhetorical. FAIR data principles define findable, accessible, interoperable, and reusable data stewardship goals (Wilkinson et al., 2016). Model cards, dataset nutrition labels, and internal algorithmic auditing provide templates for documenting model behavior, data quality, and accountability processes (Mitchell et al., 2019; Holland et al., 2018; Raji et al., 2020). These mechanisms do not create advantage automatically. They show how firms can make trust, provenance, and compliance observable enough to become part of a defensible offer.

2.7 Attribution, Data Valuation, and Machine Payments

A fifth literature cluster reframes the monetization problem. If generative AI makes imitation easier, firms and creators can respond not only by excluding use, but also by making source contribution traceable and payable. Influence-function work estimates how training examples affect model predictions (Koh & Liang, 2017). TracIn traces gradient-descent checkpoints to estimate training-data influence (Pruthi et al., 2020). TRAK scales attribution of model behavior (Park et al., 2023). DATE-LM benchmarks data-attribution methods for large language models and reports that method performance is task-sensitive, with no single method dominating across evaluated attribution tasks (Jiao et al., 2025). Deduplication research shows that repeated training data can affect memorization and behavior, which matters because similarity alone can confuse copying, memorization, and contribution (Lee et al., 2022).

Data valuation and data-economics research supplies the market-design logic. Data Shapley adapts cooperative-game valuation to estimate data-point contribution to model utility (Ghorbani & Zou, 2019). Data-as-labor proposals argue that data contributors should be treated as participants in value creation rather than as free inputs (Arrieta-Ibarra et al., 2018). Nonrival data theory explains why the same data can support multiple productive uses (Jones & Tonetti, 2020). Data-market externality research warns that uncoordinated data trade can create inefficiencies and privacy harms (Acemoglu et al., 2022). Recent AI-training-data economics work documents fragmented 2020–2025 deal structures, multiple pricing mechanisms, and frequent exclusion of original creators from compensation (Oderinwale & Kazlauskas, 2025). Together, these sources motivate a traceable contribution economy as a market-design problem rather than a solved accounting system.

Provenance and payment infrastructure provide complementary primitives. PROV-O, the W3C Provenance Ontology, defines a vocabulary for representing provenance (World Wide Web Consortium, 2013). C2PA, the Coalition for Content Provenance and Authenticity specification, defines content-credential mechanisms for digital-media provenance (Coalition for Content Provenance and Authenticity, 2025). OpenAI's Agentic Commerce Protocol (ACP) announcement, Stripe's agentic commerce documentation, Coinbase's x402 documentation for HTTP 402-style machine payments, and the Agent2Agent (A2A) interoperability specification show early infrastructure for machine-readable actions, payments, discovery, and coordination (OpenAI, 2025; Stripe, 2026b; Coinbase Developer Platform, 2026; Linux Foundation, 2026). We use these sources to define a hybrid ledger: attribution royalties for source assets plus action fees for governed service endpoints.

2.8 Agent-Accessible Service Layers

The software interface is also changing. Human-facing screens are no longer the only distribution surface. MCP is presented as an open protocol for connecting AI applications with external systems, tools, resources, and workflows (Model Context Protocol, 2026). OpenAI's Apps SDK describes a framework for building ChatGPT apps on top of MCP servers and interactive components (OpenAI, 2026a). Stripe's agentic commerce documentation describes

commerce flows where AI agents can discover, evaluate, and complete transactions; the protocol specification defines checkout operations and data structures for agent-mediated purchase flows (Stripe, 2026b;a). We do not treat these primary sources as proof of a market outcome. We treat them as evidence of a plausible new strategic layer: firms can expose controlled capabilities to agents and monetize usage, transactions, outcomes, and attributable contribution events even when UI patterns are reproducible.

2.9 Synthesis Gap

We find that the reviewed literatures do not yet fully meet. AI software-engineering research asks whether models and agents can complete programming tasks. Productivity economics asks how AI changes work, task exposure, and complementary capital. Strategy research asks which resources remain valuable when technologies diffuse. Digital innovation and ecosystem research asks how modularity and boundary resources alter value creation. IP and data-provenance research asks what can be used, copied, licensed, or governed. Attribution and data-economics research asks how contribution can be measured and valued. Platform documentation shows new agent-readable integration patterns, but it does not by itself explain competitive advantage. The commoditization question sits at the intersection of these literatures.

We interpret AI-assisted implementation as a change in the production function of software. When the production function changes, the resource basis of competition changes. A firm that previously relied on implementation difficulty may face a decline in imitation barriers. A firm that controls complementary assets may benefit because those assets become the limiting factor in the production of a trusted substitute. This synthesis also explains why measured developer productivity can be mixed while market-level commoditization still occurs. Even if an experienced maintainer is slower on a difficult task with AI, a rival, integrator, or customer may still be able to assemble an acceptable substitute for a common workflow faster than before.

The distinction is between engineering productivity and economic substitutability. Productivity measures how quickly a developer completes a defined task. Substitutability measures whether a user, buyer, or competitor can obtain a functionally acceptable alternative. AI can increase substitutability without increasing every developer's measured speed. For example, a generated workflow may be less elegant internally but good enough for a buyer whose main need is a familiar interface and a narrow integration. Conversely, a product may be easy to imitate visually but hard to substitute commercially because it controls rights, data, trust, or transactions. We therefore treat implementation commoditization as a strategic condition, not merely a coding benchmark outcome.

2.10 What Is Assembled and What Is New

Because this paper draws on several mature literatures, we state plainly which parts are inherited and which are ours, so that a reader can audit the claim rather than infer it.

Inherited. The resource-based view and its imitability conditions (Barney, 1991; Peteraf, 1993); complementary assets and appropriability regimes (Teece, 1986); switching costs and lock-in for information goods (Klemperer, 1995; Shapiro & Varian, 1998); the IT-commoditization debate (Carr, 2003; Mata et al., 1995; Wade & Hulland, 2004); layered-modular digital architecture and boundary resources (Yoo et al., 2010; Baldwin & Woodard, 2009; Ghazawneh & Henfridsson, 2013); and the empirical AI productivity and benchmark record (Peng et al., 2023; METR, 2025; Jimenez et al., 2024). None of these are our results, and none of the market statistics in Section 5 are ours: they are official statistics and published surveys, reproduced with their denominators stated.

New here. Four things. First, the identification of AI-assisted reproduction as a shift in the *appropriability regime* for software, which turns Teece's static prediction into a directional and testable one. Second, the switching condition of Section 3.3, which isolates replica cost as a term that generative capability moves independently of the other barriers, giving a concrete statement of when subscription pricing is exposed. Third, the operational coding scheme of Section 4.4 and its application to a full application built under observation, which distinguishes what is reproducible-and-cheap from what is reproducible-but-worthless-without-access. Fourth, the traceable-contribution construction of Section 3.4, which composes existing attribution, provenance and machine-payment primitives into one settlement rule with an explicit confidence gate, and states the conditions under which it should pay nothing.

Not claimed. We do not claim a causal estimate of AI's effect on reproduction cost, a population-level measurement of commoditization, or that the case generalises without further work. Section 9 states the threats to validity in full.

3 THEORY AND HYPOTHESES

3.1 DEFINITIONS

Implementation scarcity is the difficulty of reproducing a software artifact's functional behavior and user-visible workflow. Complementary scarcity is the difficulty of reproducing the assets around the artifact: data, rights, trust, compliance, distribution, network effects, operational routines, and agent-accessible service interfaces. Commoditization occurs when the market price or strategic value of implementation declines because comparable implementations become more available.

Traceable contribution is a governed record linking a source asset, rights metadata, provenance metadata, attribution evidence, and a payment endpoint to a generated output or agent action. We separate this concept from exact token-origin ownership. The narrower claim is that contribution can be estimated, bounded, audited, and paid under explicit policy when evidence confidence is sufficient.

We separate these definitions from three stronger claims: engineering is easy, every product can be legally copied, or AI always makes developers faster. The narrower claim is that AI-assisted generation can reduce reproduction cost for recognizable implementation patterns, altering the relative weights of strategic assets.

3.2 HYPOTHESES

We formulate five directional hypotheses. H1–H4 extend existing IT commoditization, RBV, IS, platform, and generative-AI strategy literatures into the specific setting of AI-assisted software reproduction; they are not claimed as wholly new empirical laws. H5 is a design and market-formation hypothesis that combines attribution, provenance, data valuation, and payment primitives. The case and market data provide mechanism evidence and plausibility checks, not definitive population-level proof.

H1: AI can erode implementation-based moats for recognizable and evaluable software patterns. When a product's distinctive value is mainly a common interface, workflow, or integration pattern, AI-assisted generation may reduce the cost of building a functional substitute. The effect depends on visibility, decomposition, repository context, evaluation quality, legal constraints, and integration friction.

H2: In AI-assisted reproduction contexts, defensible advantage shifts relatively from code artifacts alone toward controlled complementary assets. Data, content rights, trust, compliance, distribution, network position, service access, and operational capability retain stronger RBV properties than generic implementation when they remain scarce and hard to substitute.

H3: Access subscriptions face greater pricing pressure when reproducible functionality is not bundled with trust, rights, compliance, or integration depth. If customers can obtain a substitute with low switching cost and acceptable trust, then subscription access to the original implementation becomes less defensible. Monetization is therefore expected to move relatively toward usage, outcome, licensing, transaction, and service-layer pricing.

H4: Governed AI-native service layers can create monetization options that closed human-facing screens alone do not provide. Where agents become meaningful discovery, execution, or purchasing channels, agent-readable APIs, MCP servers, app tools, and commerce protocols allow firms to charge for controlled capability rather than only for human UI access.

H5: Traceable contribution and agent-action markets may partially offset software commoditization by routing revenue to source assets, generated-output contributors, and governed service/action endpoints. This requires sufficient attribution confidence, rights metadata, provenance metadata, and payment rails. H5 is constructive rather than deterministic: it identifies a plausible market design and research agenda, not a solved royalty system.

3.3 A SIMPLE MODEL

The notation is intentionally simple. Time is indexed by $t \in \mathbb{R}_{\geq 0}$. AI-assisted generation capability is $G(t) \in \mathbb{R}_{\geq 0}$, and implementation reproduction cost is $C_i(t) \in \mathbb{R}_{>0}$. Let $\mathcal{J} = \{I, D, R, T, N, K, S\}$ be the set of strategic asset classes: implementation, data, rights, trust, network or distribution position, compliance capability, and service access. For each $j \in \mathcal{J}$, $x_j(t) \in \mathbb{R}_{\geq 0}$ denotes asset strength and $\alpha_j(t) \in [0, 1]$ denotes its strategic weight, with $\sum_{j \in \mathcal{J}} \alpha_j(t) = 1$.

Assume

$$\frac{\partial C_i(t)}{\partial G(t)} < 0,$$

for tasks with visible goals, decomposable requirements, and evaluable outputs. We do not treat this assumption as universal; we scope it to recognizable implementation patterns.

Let competitive advantage be decomposed as

$$A(t) = \sum_{j \in \mathcal{J}} \alpha_j(t) x_j(t),$$

where the $x_j(t)$ terms are not assumed to share a physical unit; they represent normalized strategic asset strength within the focal market. Commoditization of implementation is modeled as

$$\frac{\partial \alpha_I(t)}{\partial G(t)} < 0,$$

while the relative weights of complementary assets increase:

$$\frac{\partial}{\partial G(t)} \sum_{j \in \mathcal{J} \setminus \{I\}} \alpha_j(t) > 0.$$

The expression does not require all complementary assets to increase for every firm. We use it to state that when implementation becomes more reproducible, the residual basis of advantage is more likely to come from complementary scarcity.

Now consider a customer deciding whether to keep paying for a subscription to an incumbent application or switch to a substitute. The structure below is not new: it is the standard switching-cost comparison of economics, in which a locked-in customer stays while the cost of moving exceeds the gain from moving (Klemperer, 1995), applied to information goods by Shapiro & Varian (1998). We add one thing to it, which is to treat the cost of *obtaining a substitute at all* as a separate term that AI-assisted generation can move independently of the others. Let $P_s, C_r, B_{sw}, B_t, B_k, B_n, \Delta V \in \mathbb{R}_{\geq 0}$, where P_s is subscription price over a decision horizon, C_r is the expected cost of obtaining or producing a replica, B_{sw} is switching cost, B_t is trust and reliability barrier, B_k is compliance or governance barrier, B_n is network or integration barrier, and ΔV is the incumbent's value premium over the substitute. Subscription defensibility weakens when

$$C_r + B_{sw} + B_t + B_k + B_n < P_s + \Delta V.$$

As AI lowers C_r, the incumbent must increase ΔV or barriers through legitimate complementary assets. The practical implication is that firms should not rely only on UI access if the UI's function can be reproduced.

3.4 Traceable Contribution Economy

We model H5 as a hybrid ledger, not as full-resolution origin tracing. Let $\mathcal{S}$ be a finite set of source assets, $\mathcal{Y}$ a finite set of generated output or agent-action events, $\mathcal{O}$ the output space, and $\mathcal{G}$ the set of governed actions. Each source asset $s \in \mathcal{S}$ publishes a ledger record

$$\ell_s = (id_s, r_s, p_s, a_s, e_s),$$

where id_s is a source identifier, r_s is rights metadata, p_s is provenance metadata, a_s is an attribution policy, and e_s is a payment endpoint. A generated event $y \in \mathcal{Y}$ records the prompt, retrieved context, model output, action call, and revenue event:

$$y = (u, X_y, o_y, g_y, R_y),$$

where u is the user or agent request, $X_y \subseteq \mathcal{S}$ is the declared context set, $o_y \in \mathcal{O}$ is the generated response or software artifact, $g_y \in \mathcal{G} \cup \{\varnothing\}$ is an optional governed action, and $R_y \in \mathbb{R}_{\geq 0}$ is gross revenue or value assigned to the event.

For each eligible source s, define a normalized attribution weight

$$w_{s,y} = \frac{\lambda\, sim(s, o_y) + (1-\lambda)\, inf(s, y)}{\sum_{k \in \mathcal{S}_y} [\lambda\, sim(k, o_y) + (1-\lambda)\, inf(k, y)]},$$

where $\mathcal{S}_y \subseteq \mathcal{S}$ is the eligible source set after rights filtering; $sim : \mathcal{S} \times \mathcal{O} \to [0,1]$ is source-output similarity; $inf : \mathcal{S} \times \mathcal{Y} \to [0,1]$ is model-influence evidence; and $\lambda \in [0,1]$ is a policy weight. The denominator is defined only

when at least one eligible source has positive evidence; otherwise, no attribution payout is assigned. The payout to source s is

$$\rho_{s,y} = \eta R_y w_{s,y} \mathbb{1}[conf_{s,y} \geq \kappa] \mathbb{1}[rights_s = 1],$$

where $\eta \in [0,1]$ is the royalty share, $conf : \mathcal{S} \times \mathcal{Y} \to [0,1]$ is attribution confidence, $\kappa \in [0,1]$ is a payout threshold, and $rights_s \in \{0,1\}$ indicates that the source permits the use under the relevant policy. If the event also calls a governed action endpoint g_y, the service provider receives an action fee $\phi(g_y) \in \mathbb{R}_{\geq 0}$, and total event settlement is

$$\mu_y = R_y - \sum_{s \in \mathcal{S}_y} \rho_{s,y} - \phi(g_y),$$

where μ_y is the residual platform, compute, and application revenue, and $\phi(g_y) = 0$ when g_y is absent. The identity is written as a decomposition of R_y rather than as a definition of it: $\rho_{s,y}$ is already a share of R_y, so an expression with R_y on both sides would be circular. Because $\sum_{s \in \mathcal{S}_y} w_{s,y} = 1$, the attribution pool is bounded above by ηR_y, and settlement is feasible with $\mu_y \geq 0$ whenever the action fee does not exceed the unattributed share, $\phi(g_y) \leq (1-\eta) R_y$. This formulation turns "access to software" into "access, action, and attribution." It also states the main limitation: if rights metadata, attribution confidence, or payment rails are weak, then $\rho_{s,y}$ should be zero or escrowed rather than asserted as an exact royalty.

4 METHODOLOGY

4.1 RESEARCH DESIGN

We use an exploratory single-case study of Myflix, a personal media application developed in the local repository through AI-assisted coding sessions. We select the case because the phenomenon is contemporary, process-oriented, and embedded in tooling, prompts, code, screenshots, and repository history. We do not estimate a population-level treatment effect. We use the case to make mechanisms visible and generate hypotheses for future multi-case or experimental work.

We triangulate repository structure, screenshots, session logs, and architecture boundaries. Repository evidence establishes that Myflix is an implemented multi-layer application rather than a static mockup. Screenshots show achieved workflows and visible product grammar. Session logs show how a broad category request became implementation plans, defect reports, and refinement loops. Architecture boundaries help distinguish generic UI reproduction from controlled service layers such as provider descriptors, local media import, playback/transcoding, account state, offline retention, subtitle matching, and confirmation-gated provider use.

4.2 CASE BOUNDARIES AND EVIDENCE HYGIENE

We bound the case to a personal media application for local media, trailers, and explicitly authorized future providers. Secrets, private tokens, credentials, and operational instructions for unauthorized acquisition are excluded. Provider evidence stays at the architectural level: controlled boundaries, explicit configuration, provenance, opt-in behavior, and confirmation-gated use.

We also avoid legal overstatement. We treat a user request for a Netflix-like experience as evidence that recognizable product categories can be specified through natural language and screenshots. We do not treat it as permission to copy trademarks, protected assets, or trade dress. The relevant scientific observation is that functional substitutes can emerge from visible patterns and iterative feedback, while legal and brand boundaries remain distinct strategic constraints.

4.3 ANALYTIC PROCEDURE

We used the following procedure:

1. Identify the theoretical constructs from literature: implementation cost, RBV resource attributes, dynamic capability, data rights, trust, compliance, and service access.
2. Extract local evidence from repository structure, TODO history, screenshots, and session prompts.
3. Classify each Myflix feature as primarily implementation-like, complementary-asset-like, or service-layer-like.
4. Compare the classification against the hypotheses.
5. Separate empirical observations from strategic interpretation and explicitly list validity threats.

4.4 Construct Operationalization

We operationalize implementation assets as visible or inspectable software elements that a skilled team can specify from behavior: page layouts, navigational structure, action buttons, API wrappers, domain models, and common background jobs. We operationalize complementary assets as elements whose value depends on control rather than description: credentials, user-specific data, content rights, lawful source access, provider contracts, operational reliability, device trust, and compliance routines. We operationalize service-layer assets as controlled capabilities exposed through programmatic interfaces: API routes, provider boundaries, authorization checks, and potential agent-readable tools.

We coded each observed feature using two questions. First, could the feature be reproduced from screenshots, prompts, code patterns, and ordinary engineering knowledge? Second, would a reproduced feature be commercially valuable without access to controlled assets? A feature such as a hero rail scores high on reproducibility and low on standalone defensibility. A feature such as profile-specific offline playback over authorized local media scores lower on standalone reproducibility because the visible button is not the asset; the asset is trusted access to the user's files, retained media, and playback state. A feature such as a provider test control sits between the two: the UI is reproducible, but the underlying trust and configuration boundary is more defensible.

Our construct coding is qualitative. We do not assign a universal numeric moat score. Instead, we use the coding to avoid treating all code as the same kind of strategic resource. The approach follows the RBV premise that resources differ in imitability and substitutability (Barney, 1991), while also following the IS literature's warning that IT artifacts gain strategic value through complementarity and organizational context (Bharadwaj, 2000; Wade & Hulland, 2004).

5 Recent Market Data and Descriptive Statistics

5.1 Data Sources and Comparability

We collected recent 2025–2026 descriptive statistics to test whether the strategic thesis is consistent with current diffusion evidence. The source base combines official enterprise statistics, an official U.S. working paper, and primary business/developer surveys. The appendix documents the data tables and chart-generation procedure.

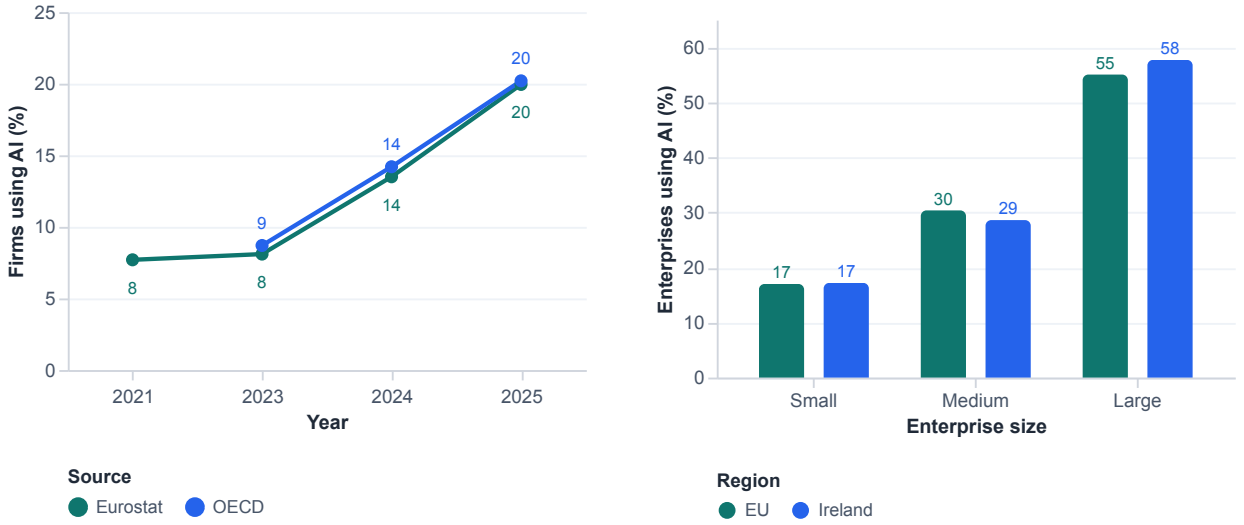
The data are not a harmonized panel. Eurostat measures EU enterprises with at least 10 employees using specified AI technologies (Eurostat, 2025). OECD reports firms using AI across countries where data are available (OECD, 2026). The U.S. Census Bureau CES working paper uses the 2026 BTOS AI supplement for U.S. employer firms excluding farms (Bonney et al., 2026; U.S. Census Bureau, 2026). Ireland's Central Statistics Office provides a national official-statistics check on the enterprise size gradient (Central Statistics Office Ireland, 2026). McKinsey and Stack Overflow add primary survey evidence on organizational and developer adoption (Singla et al., 2025; Stack Overflow, 2025). We use the sources to triangulate direction and heterogeneity, not to estimate a single global adoption rate.

5.2 Enterprise Diffusion

Figure 2, panel A, shows that official enterprise and firm measures rose quickly by 2025. Eurostat reports EU enterprise AI adoption increasing from 7.7% in 2021 and 8.1% in 2023 to 13.5% in 2024 and 19.95% in 2025 (Eurostat, 2025). OECD reports a similar movement from 8.7% in 2023 to 14.2% in 2024 and 20.2% in 2025 (OECD, 2026). We read this as evidence that AI capability is moving into firms fast enough to affect competitive dynamics. We do not read it as evidence that most firms have already transformed software production.

5.3 Firm Size and Complementary Capability

Figure 2, panel B, gives a market-level plausibility check for H2. In Eurostat's 2025 data, 55.03% of large EU enterprises used AI technologies, compared with 30.36% of medium enterprises and 17% of small enterprises. Ireland shows a similar official-statistics pattern: 57.7% large, 28.6% medium, and 17.2% small (Eurostat, 2025; Central Statistics Office Ireland, 2026). Larger firms are more likely to possess complementary data, budgets, technical staff, governance routines, procurement capacity, and integration assets. This does not prove that large firms capture more value, but it weakens a naive commoditization story in which AI instantly equalizes all firms. AI may lower implementation barriers while complementary capability still shapes adoption.



(A) Official enterprise AI adoption trend.

(B) AI adoption by enterprise size in 2025.

Figure 2: Official AI adoption trend and enterprise-size gradient. The paired panels show two market facts: official AI adoption is rising, and larger firms adopt more often. Sources: Eurostat, OECD, and Central Statistics Office Ireland (Eurostat, 2025; OECD, 2026; Central Statistics Office Ireland, 2026).

5.4 AI Use Cases and Business Functions

Figure 3 connects AI diffusion to the software-commoditization mechanism. Eurostat's most common AI technology categories in 2025 are language and content oriented: written-language analysis, picture/video/audio generation, and language or programming-code generation (Eurostat, 2025). The U.S. Census CES working paper finds that among U.S. AI-using firms, deployment is most common in sales and marketing, strategy and business development, and IT (Bonney et al., 2026). These are precisely the kinds of knowledge, interface, and workflow domains where natural-language generation, summarization, classification, and tool use can reduce implementation or coordination costs.

5.5 Measurement Contrast

Figure 4 shows why we correct deterministic claims. Official firm surveys put AI use around one-fifth in 2025 or early 2026. Employment-weighted U.S. adoption is higher at 32%, indicating that AI use is concentrated in larger employers. Management and developer surveys report much higher exposure: McKinsey reports 88% of respondent organizations regularly using AI in at least one business function, and Stack Overflow reports 84% of developers using or planning AI tools (Singla et al., 2025; Stack Overflow, 2025). The contrast is not necessarily contradictory because the denominators differ. It does, however, imply that the market is in a diffusion phase: AI is highly visible to developers and managers while economy-wide firm adoption and scaling remain uneven.

The Iceberg Index adds a complementary measurement frame: technical skill exposure can exceed visible adoption (Chopra et al., 2025). This matters for software commoditization because buyers may experience AI as a substitute-building capability before official firm-level adoption statistics show broad organizational transformation. We therefore treat adoption, exposure, displacement, and productivity as related but distinct variables.

5.6 Developer Adoption and Friction

Figure 5 adds the developer-side boundary condition. Stack Overflow reports that AI-tool use or planned use rose from 76% in 2024 to 84% in 2025, but distrust of AI-output accuracy rose from 31% to 46%, and 45% of respondents identified debugging AI-generated code as time-consuming (Stack Overflow, 2025). AI agents are not yet majority tools: 31% currently use them, 17% plan to use them, and 38% do not plan to use them. Among agent users at work, 69% report productivity increases. The combined pattern supports our conditional claim. AI development tools are diffusing quickly, but trust, debugging, evaluation, and agent maturity remain binding constraints.

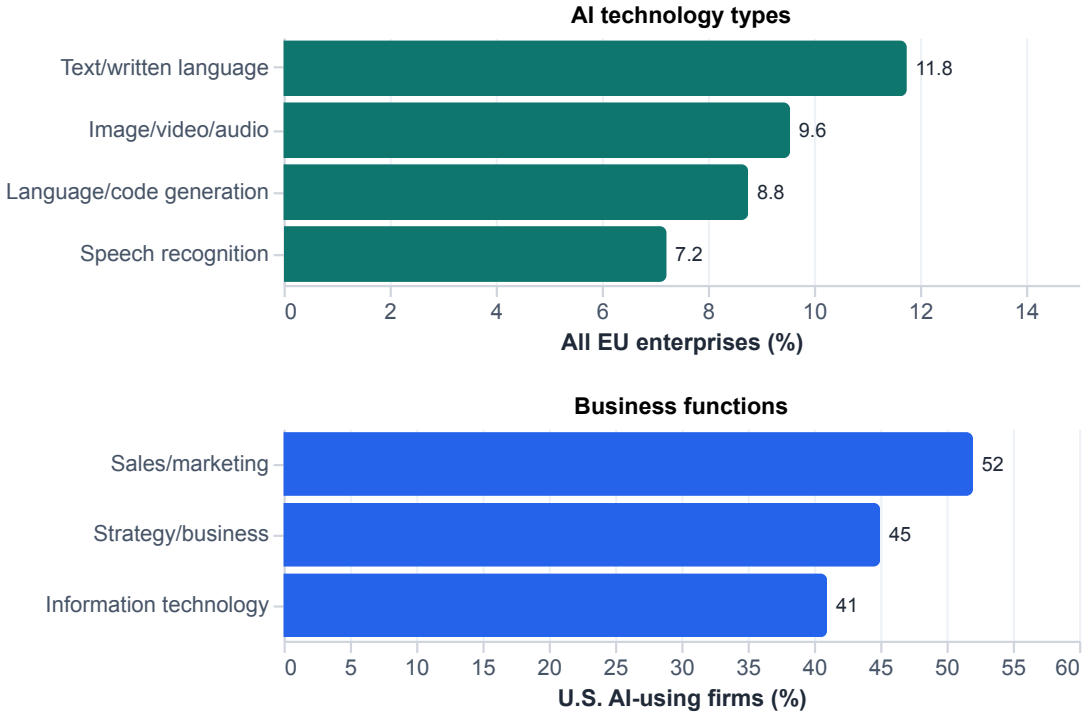


Figure 3: Selected recent AI technology and business-function measures. The panels separate denominators: Eurostat values use all enterprises; Census values use AI-using firms. The figure therefore compares patterns rather than levels. Sources: Eurostat and U.S. Census Bureau CES working paper (Eurostat, 2025; Bonney et al., 2026).

5.7 Traceable Contribution and Machine-Payment Primitives

Figure 6 synthesizes the mechanism behind H5. The figure is not a market-size estimate. It is a source-backed value-stack representation that links scholarly attribution methods, provenance standards, and emerging machine-payment protocols (Longpre et al., 2024; Koh & Liang, 2017; Pruthi et al., 2020; Park et al., 2023; Jiao et al., 2025; World Wide Web Consortium, 2013; Coalition for Content Provenance and Authenticity, 2025; OpenAI, 2025; Stripe, 2026b; Coinbase Developer Platform, 2026; Linux Foundation, 2026). The sequence begins with source assets and rights metadata, moves through provenance and attribution evidence, records generated responses or agent actions as revenue events, and settles value through creator royalties and governed service/action fees. The figure abbreviates W3C PROV-O and C2PA as PROV/C2PA, attribution methods as TRAK/DATE-LM, OpenAI Apps SDK and Agentic Commerce Protocol as Apps/ACP, and Stripe agentic commerce plus Coinbase x402 payment rails as Stripe/x402.

Recent consumer-surplus evidence motivates the market-design problem. Brynjolfsson et al. (2026) estimate that U.S. consumer surplus from generative-AI chatbots rose from $116 billion to $172 billion between 2025 and early 2026, while mean monthly willingness-to-accept for giving up access rose from $98 to $124.50. That does not prove that source contributors can capture the surplus. It shows that generated responses create meaningful consumer value that is not fully reflected in provider revenue. A traceable contribution economy asks whether some of that value can be routed to the source assets, software services, and governed action endpoints that make generated outputs useful.

The market data refine the hypotheses. They are consistent with H1 as a directional pressure because adoption is rising rapidly and developer exposure is high, but they do not measure implementation-reproduction cost directly. They are consistent with H2 because firm size, sector, and employment weighting remain important. They qualify H3 because pricing pressure will be uneven: customers with high trust, compliance, or integration needs may keep paying for incumbents even when generic functionality becomes easier to reproduce. They make H4 plausible as a response rather than as a proven outcome: management-survey evidence shows growing agent experimentation, but current official data do not yet measure agent-service monetization directly. They motivate H5 because high consumer value, provenance infrastructure, attribution methods, and machine-payment protocols now coexist; they do not yet demonstrate that legally robust token-level settlement has arrived.

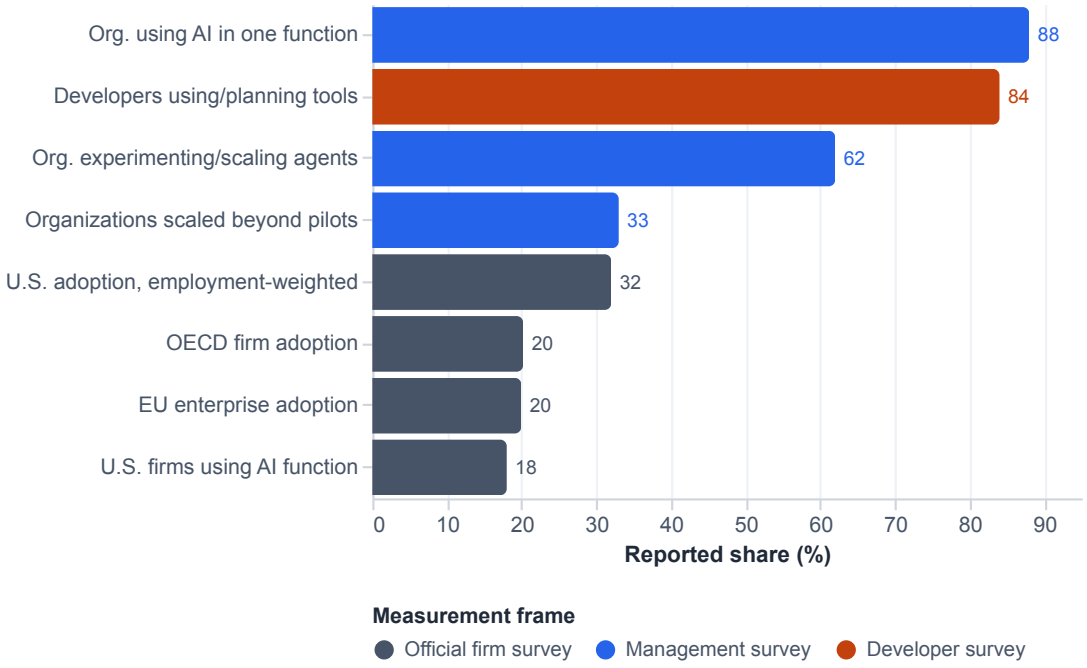


Figure 4: Recent AI adoption statistics by measurement frame. The wide spread cautions against treating AI adoption as a single market number. Sources: Eurostat, OECD, U.S. Census Bureau, McKinsey, and Stack Overflow (Eurostat, 2025; OECD, 2026; Bonney et al., 2026; Singla et al., 2025; Stack Overflow, 2025).

6 Case Study: Myflix

6.1 Case Overview

Myflix is a browser-first personal media application. Its implementation spans a TypeScript monorepo with a Next.js web app, Fastify API, background worker, domain contracts, media utilities, provider clients, and shared UI package. The app includes account/profile flows, a media-forward home surface, detail pages, watch playback, downloads, offline retention, profile preferences, TV Mode, provider settings, subtitle controls, and local deployment routines.

Myflix development began from a recognizable category request: build an application similar to a premium streaming service, with highlighted banners, shows, clicking through content, search, offline downloads, storage/NAS settings, and a polished interface. Subsequent prompts converted the broad request into implementation plans and then into bug-fix and refinement loops. We did not observe a single generated artifact. We observed an iterative process of planning, editing, running checks, inspecting screenshots, and refining behavior.

6.2 Implementation Scope

The implementation demonstrates more than a static mockup. The web app provides browsing, detail, watch, settings, downloads, and TV Mode surfaces. The API handles persistence, authentication, profile-aware state, provider operations, media byte serving, and acquisition/offline job state. The worker supports background operations. Shared packages define domain contracts, media logic, provider boundaries, and UI primitives.

We use this scope to separate implementation reproduction from product defensibility. The visible browsing experience shows that a recognizable software category can be implemented through iterative AI-assisted work. The surrounding architecture shows why implementation is not the whole product. Playback reliability, local file handling, credentials, provider terms, data provenance, and deployment routines remain significant constraints.

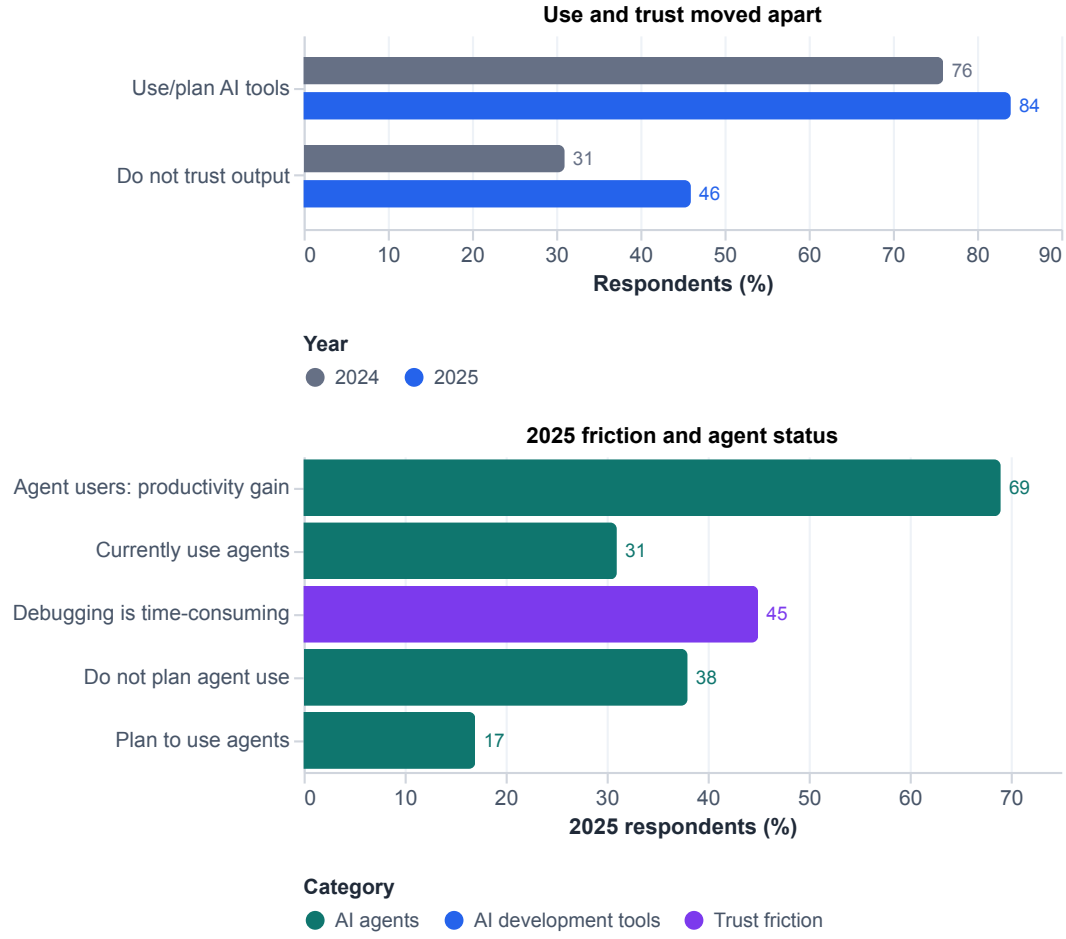


Figure 5: Developer AI adoption and friction in Stack Overflow's 2025 Developer Survey. The upper panel separates 2024 and 2025 values for AI-tool use and trust; the lower panel shows 2025 agent use, intent, reported productivity, and debugging friction (Stack Overflow, 2025).

6.3 Qualitative UI Evidence

The screenshots show several relevant interface achievements. The home screenshot contains a full-bleed media hero, navigation, search, playback actions, offline/download affordances, and genre rails. The detail screenshot contains season tabs, episode rows, per-episode Play/Offline/Download controls, and bulk season actions. The watch-player screenshot contains a timeline, player controls, episode context, and offline state. The TV Mode screenshots contain an on-now card and upcoming schedule rows. The downloads and browse screenshots show retained media management, Movies and Series rails, hover/focus previews, and dense navigation. The settings screenshots contain diagnostics, provider cards, confirmation-gated controls, and explicit authorized-source boundaries.

Our screenshot evidence supports the claim that visible product grammar can be reproduced through iterative AI-assisted development. The term product grammar refers to layout structure, control vocabulary, navigation pattern, and user expectation. It does not imply copying protected expression. The Myflix screenshots demonstrate a recognizable consumer-media grammar: hero selection, rails, detail panels, episode actions, and provider settings. Such grammar is easier to describe and inspect than hidden operational assets.

6.4 Prompt and Process Evidence

Our session evidence shows a pattern of progressive specification. The initial request specified a high-level category and desired experience. A later implementation-plan prompt decomposed the app into web, API, worker, domain,

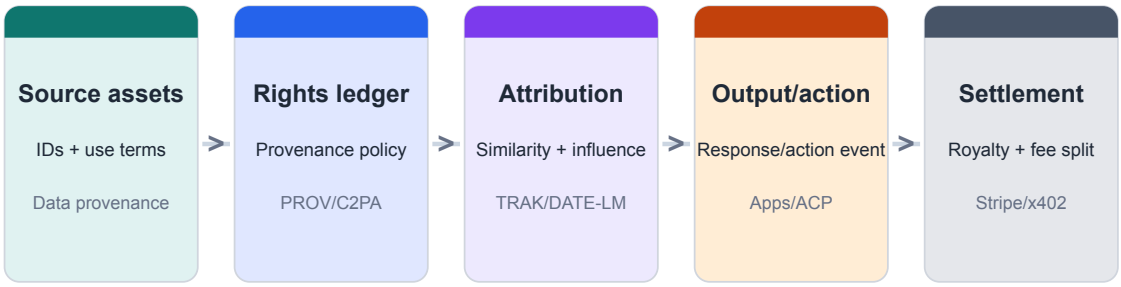


Figure 6: Traceable AI value stack. The mechanism links rights-aware source assets, provenance metadata, attribution evidence, generated outputs or actions, and settlement rails. Abbreviations refer to provenance standards (PROV-O and C2PA), attribution methods (TRAK and DATE-LM), OpenAI app and commerce interfaces (Apps SDK and ACP), and payment infrastructure (Stripe agentic commerce and Coinbase x402).

media, provider, and UI layers. Subsequent prompts addressed concrete defects and new features: player controls, hover previews, timeline jumping, sparse library content, Series/Movies navbar filters, duplicate genre rails, offline-only Series browsing, disabled offline controls, download progress, subtitles, detail back behavior, Picture-in-Picture handoff, and large-season performance.

We interpret this process evidence as a rejection of the simple "prompt once and receive a product" narrative. Our evidence suggests an interactive production function:

$$Q = h(P, E, R, V, J),$$

where Q is achieved product quality, P is prompt specificity, E is existing repository context, R is runtime and validation feedback, V is visual inspection, and J is human judgment. AI lowers the marginal cost of turning feedback into implementation, but the feedback loop still matters.

6.5 Service-Layer and Provider Evidence

The most strategically interesting parts of Myflix are not the visible rails or buttons. They are the controlled boundaries around media, providers, and local state. The codebase separates provider descriptors and clients from UI presentation. It distinguishes local files, configured metadata sources, subtitle sources, and explicit provider settings. It uses confirmation-gated controls for provider enablement and testing. It maintains profile-aware state, offline retention, playback progress, and local availability.

We interpret this architecture as consistent with H2 and as a mechanism example for H4. A browsing UI is imitable; a governed set of data rights, user configuration, provider contracts, local file provenance, and reliable playback operations is harder to reproduce. Similarly, an agent-readable service layer could expose controlled operations such as search, entitlement checks, metadata retrieval, transaction initiation, or content delivery without relying only on human UI access. We use the case to illustrate a strategic distinction between the screen as interface and the service boundary as asset.

7 Results

7.1 Result 1: Implementation Reproduction Was Feasible, But Not Automatic

The Myflix case is consistent with H1, within its narrow scope. A recognizable application category was transformed into a working implementation through AI-assisted development. The architecture and screenshots show depth beyond a landing page: browsing, playback, persistence, provider boundaries, offline behavior, and validation. Our evidence suggests that implementation cost can fall materially for visible, decomposable, feedback-rich product grammar.

At the same time, our evidence contradicts the strongest automation narrative. The process required many iterations, bug reports, screenshots, local checks, and design refinements. The AI agent did not replace product judgment or

validation. It accelerated a feedback loop. We read this as consistent with the mixed literature: AI can help in bounded settings (Peng et al., 2023), agent systems can solve some repository tasks (Jimenez et al., 2024; Yang et al., 2024), but experienced developers can also be slowed in difficult real settings (METR, 2025). The result is not "AI builds software for free." It is "AI can reduce the marginal cost of implementing and revising recognizable patterns when feedback is available."

7.2 Result 2: The UI Shell Was Less Defensible Than the Controlled Assets

The case is consistent with H2. The media browsing shell is the most visible and most reproducible layer. The more defensible layers are less visible: local media files, lawful access, profile data, provider configuration, credentials kept outside source control, playback/transcoding logic, offline retention, user trust, and explicit provenance boundaries. These assets match RBV logic better than generic UI implementation. They are more likely to be rare, imperfectly imitable, and institutionally embedded (Barney, 1991; Wade & Hulland, 2004).

We observe this distinction in the settings screenshot and provider architecture. Provider controls are not merely UI widgets; they encode authorization, configuration, testing, and source provenance. Similarly, playback and offline behavior require operational reliability. A competitor can imitate a button labeled Download more easily than it can reproduce lawful content rights, user-specific libraries, trusted provider relationships, and reliable device-level offline semantics.

7.3 Result 3: Monetization Pressure Depends on Switching, Trust, and Rights

We treat H3 as theoretically motivated and case-plausible, not directly market-proven. Myflix is not a commercial SaaS pricing experiment. It does, however, illustrate why subscription access to generic functionality may face pressure. If users can describe a desired workflow and obtain a functional substitute, then the willingness to pay for access to the original workflow depends increasingly on complementary value. For media software, this means licensed content, recommendation data, household history, device support, reliability, trust, and compliance. For enterprise SaaS, the analogous assets may be workflow data, audit trails, integrations, procurement trust, certifications, and embedded process change.

The monetization condition from the model provides the general logic. If C_r falls, the incumbent must increase ΔV or the legitimate barriers B_{sw}, B_t, B_k, and B_n. Otherwise, subscription access to reproducible app functionality becomes less defensible. That does not eliminate subscriptions. It changes what subscriptions must include: not just screens, but trusted operations, data, support, compliance, rights, and integration depth.

7.4 Result 4: Agent-Accessible Services Are a Strategic Response

The related work and service-boundary interpretation make H4 plausible. MCP, OpenAI Apps SDK, and agentic commerce documentation indicate that software capabilities can be exposed through agent-readable tools and transaction protocols (Model Context Protocol, 2026; OpenAI, 2026a; Stripe, 2026b). That matters because an AI agent does not need a copied UI if it can call a controlled service. Firms may therefore defend value by becoming the authoritative service for data, rights, transactions, or outcomes.

In the Myflix analogy, a human-facing browsing UI is only one access layer. A more defensible architecture would expose controlled, policy-aware operations: search the user's authorized library, resolve playable files, check offline availability, retrieve subtitles, schedule playback, or initiate a licensed transaction. The same principle applies to other domains: tax filing, insurance claims, procurement, travel booking, analytics, and compliance workflows. The monetizable asset becomes a governed capability endpoint.

7.5 Result 5: Market Data Support Diffusion, Not Determinism

The descriptive market data are consistent with the direction of our thesis but correct its strongest version. Official statistics show rapid AI diffusion, but not universal adoption. Eurostat and OECD place 2025 firm or enterprise AI adoption around 20%, while the U.S. Census BTOS supplement places firm-level use at 18% and employment-weighted use at 32% in the November 2025–January 2026 reference period (Eurostat, 2025; OECD, 2026; Bonney et al., 2026). These values imply that AI is now material enough to affect software markets, but they do not imply that every buyer can already replace every vendor.

The size gradient matters most for interpretation. Large EU enterprises report more than triple the AI adoption rate of small enterprises, and the Irish official data show the same pattern (Eurostat, 2025; Central Statistics Office Ireland,

Table 1: Qualitative classification of selected Myflix features. The table separates visible implementation elements from data, trust, operational, rights, and service-boundary assets that are harder to substitute.

Feature class	Primary asset type	Strategic interpretation
Home hero, rails, hover previews	Implementation	Highly visible and therefore more reproducible from prompts and screenshots.
Season tabs and episode controls	Implementation plus workflow state	UI is reproducible; durable value depends on accurate episode data and progress state.
Profile-aware watch progress	Data and trust	Requires user-specific history, persistence, and reliable identity boundaries.
Offline retention and downloads	Operational capability	Button is reproducible; trusted local availability and storage behavior are harder.
Provider settings and tests	Service boundary	UI is simple; provenance, authorization, and configuration rules carry the value.
Playback/transcoding decisions	Operational capability	Requires device knowledge, media probing, and reliable stream behavior.
Potential agent tool interface	Service access	Monetizable if it exposes governed search, playback, licensing, or transaction operations.

2026). We interpret this as market-level consistency with H2, not proof of value capture: AI tools do not eliminate the importance of complementary assets. Instead, firms with stronger data, talent, governance, and integration capacity may be better positioned to exploit AI and to defend the service layers built around it.

Developer data add a second correction. Stack Overflow's 2025 survey shows high AI-tool exposure, but also high distrust and debugging friction (Stack Overflow, 2025). That pattern is consistent with the Myflix process evidence: the agent accelerated iteration, but the work still required tests, screenshots, quality control, and human judgment. We therefore treat commoditization as a pressure on implementation scarcity, not as a claim that software development has become frictionless.

7.6 Result 6: Traceable Contribution Markets Are Plausible But Not Solved

We treat H5 as a design implication rather than as an observed Myflix revenue result. The Myflix analogy clarifies the problem. A model can help a user generate a media application inspired by a familiar category, but the economic system has few mature mechanisms for compensating the upstream content, interface knowledge, code examples, documentation, and service endpoints that made the generated artifact useful. Defensive IP alone creates a binary fight over permission. A traceable contribution ledger creates a second path: source owners can publish machine-readable terms, models and agents can record context and action events, attribution methods can produce confidence-weighted contribution evidence, and payment rails can settle royalties or action fees when policy allows.

This result is intentionally bounded. Current attribution methods are approximate, task-sensitive, and often unavailable for closed frontier models (Jiao et al., 2025). Provenance standards record lineage but do not determine legal rights or prices (World Wide Web Consortium, 2013; Coalition for Content Provenance and Authenticity, 2025). Agentic commerce and x402-style payment rails show machine-payment primitives, not a settled creator-compensation economy (Stripe, 2026b; Coinbase Developer Platform, 2026). We therefore frame H5 as a constructive research and market-design direction: an AI-first economy can monetize access, action, and attribution, but only if confidence, consent, rights metadata, and settlement governance are strong enough.

7.7 Result 7: Feature Classification

Table 1 summarizes our qualitative coding of selected Myflix features. The classification is not a legal opinion and not a universal ranking. It is a case-study coding exercise that separates visible implementation from harder-to-substitute complements.

Our coding supports the central thesis. The most obvious parts of the product are the least defensible by themselves. The less obvious parts are closer to strategic assets because they require data, trust, rights, or operational control. We

expect this pattern in many software categories. A project-management board, an analytics chart, or an insurance-claim workflow can be recreated at the screen level; the harder assets are historical records, institutional approvals, compliance evidence, user trust, integrations, and the right to act.

Our feature classification also clarifies why AI-first commoditization can coexist with successful software businesses. A firm does not need implementation to remain scarce if it owns the service boundary that substitutes need. The strategic task is to move from selling a screen to selling a reliable capability. In the Myflix case, that capability would be lawful media organization and playback over user-controlled sources. In enterprise software, it might be approved spend execution, regulated workflow evidence, or a trusted system of record.

8 Discussion

8.1 From Software as Artifact to Software as Capability

We derive the main strategic implication as a shift from software as artifact to software as capability. When implementation is scarce, the artifact itself can be a moat. When implementation becomes more reproducible, the artifact becomes a delivery vehicle for deeper assets. Competitive advantage then depends on who controls the inputs, rights, data, trust, distribution, compliance, and service endpoints.

That reframing also changes how firms should interpret AI competition. A firm threatened by AI clones should ask which parts of its product are merely visible implementation patterns and which parts are protected by complementary scarcity. If the answer is mostly visible implementation, the firm is vulnerable. If the answer includes proprietary data, trusted workflows, regulated operations, exclusive rights, and agent-accessible services, the firm has stronger defenses.

8.2 IP Protection in an AI-First Economy

IP protection becomes both more important and less sufficient. Copyright, trademark, patent, trade secret, contract, and database rights can constrain copying, but they do not automatically create product-market defensibility. A firm may prevent literal copying of assets or code while still facing substitutes that reproduce functional value through different implementation. Conversely, open interfaces can increase distribution and ecosystem value, as API interoperability debates show (Supreme Court of the United States, 2021).

We draw a practical IP lesson: protect what remains scarce. For consumer media, this may be content rights, brand trust, recommendation data, and distribution relationships. For enterprise SaaS, it may be customer data models, compliance certifications, workflow integrations, audit trails, and procurement trust. For AI-native businesses, it may be consented datasets, model evaluation data, tool-use telemetry, and agent-service contracts. Copyright claims against generated substitutes may be part of a defense, but they are unlikely to be the only defense.

8.3 Monetization Beyond Human-Facing SaaS

Our monetization thesis is not that SaaS subscriptions disappear. Subscription pricing tied only to access to reproducible screens becomes weaker. Durable pricing needs one or more of the following:

- scarce content or data access;
- legal rights or licenses;
- trusted execution and compliance;
- workflow embedding and switching costs;
- network or marketplace position;
- usage-based or outcome-based value capture;
- transaction fees through agentic commerce;
- attribution royalties for source assets;
- service-layer licensing to other apps or agents.

Agentic commerce and MCP-like interfaces make this shift more concrete. If an AI assistant can act on behalf of a user, then the assistant needs reliable services, permissions, prices, provenance, and transaction semantics. Firms that expose controlled capabilities may monetize at the point of action. Firms that only hide value behind a human UI may find the UI bypassed, summarized, or replicated.

8.4 From IP Defense to Attribution-Based Monetization

We propose attribution-based monetization as a complement to IP defense. Defensive IP asks whether a model builder, software imitator, or downstream user may use a protected asset. That question remains necessary. It is not sufficient for an AI-first economy because users increasingly want individualized software, integrated tools, and generated responses assembled from many sources. A pure exclusion model can block use, but it does not easily price partial contribution when a generated output draws on many documents, code examples, content items, APIs, and service calls.

The traceable contribution ledger reframes the problem as governed participation. A source owner can provide software, documentation, content, data, or service descriptions to model providers and agent ecosystems under machine-readable terms. The ledger records provenance, rights, attribution policy, and payment endpoint. When a response, code patch, workflow, or agent action is generated, the system records the output/action event and computes a confidence-weighted attribution share. The payout can combine a micro-royalty for source contribution with an action fee for a governed service endpoint. This is the economic version of the Myflix distinction: the visible media-app grammar may be reproducible, but controlled assets and authoritative actions can still be paid.

The mechanism also changes incentives for model providers. Instead of treating training-data owners only as legal adversaries, providers could offer a participation path: contribute high-quality software, content, data, tests, schemas, or service APIs; receive usage telemetry and micro-settlement when attributable outputs or actions create value. Such a system would not eliminate litigation because rights disputes, market power, and privacy risks remain. It would, however, create a positive-sum channel for lawful contribution where current data deals are fragmented and often exclude original creators (Oderinwale & Kazlauskas, 2025).

We do not claim exact token royalties are solved. Similarity can over-credit common phrases or public patterns. Influence methods require task-specific validation and model access. Provenance chains can be incomplete. Payment fees can exceed micro-royalty value. The scientific contribution is narrower: if software commoditization shifts value from artifacts toward generated use, then the economic layer should trace and price contribution to generated outputs and agent actions, not only sell access to static applications.

8.5 Implications for Software Builders

We derive five practical design principles from the case.

First, separate generic implementation from scarce assets. Treat screens as replaceable and invest in data quality, authorization, provenance, and service contracts.

Second, build explicit service boundaries. APIs, MCP servers, app tools, and event interfaces can become distribution channels for agents.

Third, design for trust and compliance early. If AI reduces implementation cost for everyone, trust becomes a stronger differentiator.

Fourth, use AI to accelerate iteration, but preserve evaluation. Tests, screenshots, user review, and operational checks remain necessary.

Fifth, avoid confusing inspiration with lawful copying. Functional categories can be reproduced, but brands, trademarks, protected assets, and contractual data access require independent rights.

8.6 Market-Level Implications

Recent statistics imply a market in transition rather than a market already commoditized. Official 2025 and early-2026 measures place firm adoption near one-fifth, with higher adoption when weighted by employment and much higher adoption among large or knowledge-intensive firms (Eurostat, 2025; OECD, 2026; Bonney et al., 2026). The market-level implication is therefore uneven pressure. Common features and visible workflows face faster imitation first. Regulated, trust-heavy, data-rich, or deeply integrated workflows face slower substitution because the complementary assets remain scarce.

For incumbent SaaS firms, feature parity will arrive faster in the parts of the product that are visible, common, and easy to evaluate. The marginal feature gap between an incumbent and a new entrant may narrow when entrants can generate common workflows, UI states, integration code, and tests. Incumbents should therefore avoid treating feature count as the main moat. A more robust strategy is to make the product the trusted place where data, approvals, policies, transactions, and audit evidence live.

For startups, our implication is two-sided. AI lowers the cost of entering a software category, but it also lowers the cost for followers to imitate the startup. The official size-gradient data caution against assuming that small firms automatically capture the full benefit: large firms report much higher adoption, likely because they have more complementary resources. A startup that uses AI only to build screens may face rapid competition. A startup that uses AI to reach a proprietary data position, secure distribution, build a trusted workflow, or expose a valuable service endpoint may turn the same AI capability into leverage.

For buyers, our implication is bargaining power with a trust constraint. If a workflow can be reproduced or self-hosted at lower cost, buyers may resist high subscription prices for generic interfaces. However, Stack Overflow's developer evidence and OpenAI's benchmark critique both point to the same caution: generated software still requires validation, debugging, and trustworthy evaluation (Stack Overflow, 2025; OpenAI, 2026b). Buyers will still pay for reliability, compliance, support, integration, and risk reduction. "We have the feature" becomes less persuasive than "we are the trusted system that can perform the action, prove it, and carry the risk."

For platforms, our implication is that control points may move upward. The platform that brokers agent access, identity, payment, permission, and service discovery can capture value even when individual app screens commoditize. MCP-style tools, app components, and agentic payment protocols are early examples of this movement. McKinsey's 2025 survey reports broad experimentation with AI agents, but official statistics do not yet measure agent-service monetization directly (Singla et al., 2025). The strategic question becomes who owns the registry, trust layer, policy enforcement, and transaction relationship.

For consumers and creators, traceable contribution markets change the default bargain. Consumers can assemble individualized software and workflows around their own needs while still consuming governed services and licensed content. Creators, developers, and data providers can choose to expose assets to model and agent ecosystems when attribution, rights, and payment rules are acceptable. Agent-to-agent communication amplifies this possibility because economic events can occur between software actors: one agent requests a data transformation, another calls a licensed service, a third settles a payment, and the ledger records source contribution and action fees (Linux Foundation, 2026; Coinbase Developer Platform, 2026). The market-level implication is a possible move from app-store economies of scale toward token-response and action-event economies of scale.

9 Limitations and Threats to Validity

We acknowledge several limitations.

First, we rely on a single case. Myflix is useful for mechanism discovery but cannot estimate average AI productivity or market-wide commoditization. Future work should compare multiple products, domains, and teams.

Second, our case is non-randomized. There is no matched non-AI development counterfactual. Repository evidence shows implemented scope, not causal acceleration.

Third, the author and AI agent interacted iteratively. Human judgment, prompt quality, and existing tooling may explain part of the outcome. We interpret the case as AI-assisted production, not autonomous production.

Fourth, the product category is visually recognizable and feedback-rich. AI may be less effective in domains where requirements are tacit, safety-critical, mathematically subtle, or hard to evaluate.

Fifth, legal and ethical constraints limit imitation. Trademarked brands, copyrighted assets, proprietary data, and unauthorized content sources remain outside lawful imitation. Those assets become more important as generic implementation becomes easier to reproduce.

Sixth, source documentation for MCP, Apps SDK, and agentic commerce reflects platform directions at the time of writing. Standards, APIs, and commercial adoption may change.

Seventh, the market statistics are descriptive and non-harmonized. Eurostat, OECD, U.S. Census, McKinsey, and Stack Overflow use different populations, denominators, survey frames, and definitions. We use them to show diffusion, heterogeneity, and friction; we do not treat them as a single comparable adoption panel.

Eighth, enterprise AI adoption does not directly measure software commoditization. A firm can use AI for marketing, strategy, or document analysis without using AI to reproduce software. Conversely, developer AI usage can increase substitutability for some software patterns before it appears in official firm-level adoption statistics.

Ninth, recent market data will change. Future revisions should refresh the official statistics, regenerate the figures, and update any claims whose source definitions changed.

Tenth, traceable contribution markets remain technically and institutionally immature. Training-data attribution is approximate, model- and task-sensitive, and often unavailable without model checkpoints or privileged logs. Similarity can misread common expression as contribution, while influence scores can be expensive or unstable. Provenance metadata can be missing, forged, or stripped. Rights metadata can conflict across jurisdictions. Micro-payments can fail when transaction cost exceeds royalty value. We therefore treat H5 as a bounded mechanism and research agenda, not as a claim that legally exact token-level royalties are currently solved.

10 Conclusion

AI-assisted software generation changes the economics of implementation but does not abolish strategy, law, or engineering. Our Myflix evidence shows that a recognizable consumer software category can be built and refined rapidly through iterative AI-assisted development. Recent 2025–2026 market data show the same directional pressure at a broader level: AI adoption is rising quickly, but it remains uneven across firm size, sector, measurement frame, and developer trust. Our interpretation is therefore bounded: AI can reduce reproduction cost for visible, decomposable, evaluable implementation patterns. As a result, implementation-only moats weaken first where workflows are common, observable, and easy to evaluate.

The strategic response is not to deny commoditization, but to move the basis of advantage. Firms should invest in assets that remain scarce: data, rights, trust, compliance, distribution, network position, operational reliability, agent-accessible service layers, and traceable contribution records. Monetization should follow the same shift. When screens are reproducible, charging for screens alone is fragile. Charging for trusted capability, licensed access, transactions, outcomes, governed agent services, and attributable source contribution is more defensible. The empirical task ahead is to measure where this shift occurs fastest, which complementary assets slow it down, how agent-service interfaces change the revenue mix of software businesses, and whether attribution/payment ledgers can route value to source contributors without overstating causal contribution.

Author Contributions

Marius-Constantin Dinu conceived the study, defined the research question, developed the core hypotheses, and authored the scientific framing of software commoditization, complementary assets, agent-service monetization, and traceable contribution markets. The author originated and guided the Myflix case, supplied the case context, selected the local evidence base, provided screenshots and development-history evidence, directed the Codex session analysis, and set the ethical and legal boundaries for what could be used as evidence. The author also orchestrated the agent-assisted research workflow, reviewed intermediate outputs, corrected the direction of the argument, refined the hypotheses, and determined which empirical observations, market statistics, mathematical model elements, and literature clusters were relevant to the final paper. AI assistance was used as a supervised research and drafting tool for literature organization, source-note consolidation, local evidence synthesis, figure generation, LaTeX editing, and consistency checks. Final responsibility for the argument, interpretation, evidence selection, and manuscript content remains with the author.

Acknowledgments

The author acknowledges the open research, legal, and technical documentation cited here and the local Myflix development history used as an exploratory case-study record.

References

Daron Acemoglu, Ali Makhdoumi, Azarakhsh Malekian, and Asuman Ozdaglar. Too much data: Prices and inefficiencies in data markets. *American Economic Journal: Microeconomics*, 14(4):218–256, 2022. doi: 10.1257/mic.20200200.

Ron Adner. Ecosystem as structure: An actionable construct for strategy. *Journal of Management*, 43(1):39–58, 2017. doi: 10.1177/0149206316678451.

Imanol Arrieta-Ibarra, Leonard Goff, Diego Jiménez-Hernández, Jaron Lanier, and E. Glen Weyl. Should we treat data as labor? moving beyond "free". *AEA Papers and Proceedings*, 108:38–42, 2018. doi: 10.1257/pandp.20181003.

Pierre Azoulay, Joshua L. Krieger, and Abhishek Nagaraj. Old moats for new models: Openness, control, and competition in generative ai. NBER Working Paper 32474, National Bureau of Economic Research, 2024. Accessed 2026-05-07.

Carliss Y. Baldwin and C. Jason Woodard. The architecture of platforms: A unified view. In Annabelle Gawer (ed.), *Platforms, Markets and Innovation*, pp. 19–44. Edward Elgar, 2009.

Shraddha Barke, Michael B. James, and Nadia Polikarpova. Grounded copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA1):85–111, 2023. doi: 10.1145/3586030.

Jay Barney. Firm resources and sustained competitive advantage. *Journal of Management*, 17(1):99–120, 1991. doi: 10.1177/014920639101700108.

Emily M. Bender and Batya Friedman. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 2018. doi: 10.1162/tacl_a_00041.

Anandhi Bharadwaj, Omar A. El Sawy, Paul A. Pavlou, and N. Venkatraman. Digital business strategy: Toward a next generation of insights. *MIS Quarterly*, 37(2):471–482, 2013. doi: 10.25300/MISQ/2013/37:2.3.

Anandhi S. Bharadwaj. A resource-based perspective on information technology capability and firm performance: An empirical investigation. *MIS Quarterly*, 24(1):169–196, 2000. doi: 10.2307/3250983.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, et al. On the opportunities and risks of foundation models. https://arxiv.org/abs/2108.07258, 2021.

Kathryn Bonney, Cory L. Breaux, Emin Dinlersoz, Lucia S. Foster, John C. Haltiwanger, and Aditya A. Pande. The microstructure of ai diffusion: Evidence from firms, business functions, and worker tasks. Working Paper CES-26-25, U.S. Census Bureau, Center for Economic Studies, 2026.

Erik Brynjolfsson, Daniel Rock, and Chad Syverson. Artificial intelligence and the modern productivity paradox: A clash of expectations and statistics. Working Paper 24001, National Bureau of Economic Research, 2017.

Erik Brynjolfsson, Danielle Li, and Lindsey R. Raymond. Generative ai at work. *The Quarterly Journal of Economics*, 140(2):889–942, 2025. doi: 10.1093/qje/qjae044.

Erik Brynjolfsson, Avinash Collis, Felix Eggers, Sophia Kazinnik, and David Nguyen. What is generative ai worth? Technical report, Stanford Digital Economy Lab, 2026. Working paper posted 2026-04-13; accessed 2026-05-07.

Nicholas G. Carr. It doesn't matter. *Harvard Business Review*, 81(5):41–49, 2003.

Central Statistics Office Ireland. Information society statistics – enterprises 2025: Artificial intelligence. https://www.cso.ie/en/releasesandpublications/ep/p-isse/informationsocietystatistics-enterprises2025/artificialintelligence/, 2026. Released 2026-02-06; accessed 2026-05-07.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. https://arxiv.org/abs/2107.03374, 2021.

Henry Chesbrough and Richard S. Rosenbloom. The role of the business model in capturing value from innovation: Evidence from xerox corporation's technology spin-off companies. *Industrial and Corporate Change*, 11(3):529–555, 2002. doi: 10.1093/icc/11.3.529.

Ayush Chopra, Santanu Bhattacharya, DeAndrea Salvador, Ayan Paul, Teddy Wright, Aditi Garg, Feroz Ahmad, Alice C. Schwarze, Ramesh Raskar, and Prasanna Balaprakash. The iceberg index: Measuring skills-centered exposure in the ai economy. https://arxiv.org/abs/2510.25137, 2025. Accessed 2026-05-07.

Clayton M. Christensen, Mark W. Johnson, and Darrell K. Rigby. Foundations for growth: How to identify and build disruptive new businesses. *MIT Sloan Management Review*, 43(3):22–31, 2002.

Eric K. Clemons and Michael C. Row. Sustaining it advantage: The role of structural differences. *MIS Quarterly*, 15 (3):275–292, 1991. doi: 10.2307/249639.

Coalition for Content Provenance and Authenticity. C2PA technical specification, version 2.2. https://spec.c2pa.org/specifications/specifications/2.2/specs/C2PA_Specification.html, 2025. Accessed 2026-05-07.

Coinbase Developer Platform. x402: The internet-native payments protocol. https://docs.cdp.coinbase.com/x402/welcome, 2026. Accessed 2026-05-07.

Ingemar Dierickx and Karel Cool. Asset stock accumulation and sustainability of competitive advantage. *Management Science*, 35(12):1504–1511, 1989. doi: 10.1287/mnsc.35.12.1504.

Marius-Constantin Dinu. Parameter choice and neuro-symbolic approaches for deep domain-invariant learning. https://arxiv.org/abs/2410.06235, 2024. Doctoral thesis. Accessed 2026-05-07.

Marius-Constantin Dinu, Claudiu Leoveanu-Condrei, Markus Holzleitner, Werner Zellinger, and Sepp Hochreiter. Symbolicai: A framework for logic-based approaches combining generative models and solvers. In Vincenzo Lomonaco, Stefano Melacci, Tinne Tuytelaars, Sarath Chandar, and Razvan Pascanu (eds.), *Proceedings of The 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 869–914. PMLR, 2025. Accessed 2026-05-07.

Ben Eaton, Silvia Elaluf-Calderwood, Carsten Sørensen, and Youngjin Yoo. Distributed tuning of boundary resources: The case of apple's ios service system. *MIS Quarterly*, 39(1):217–243, 2015. doi: 10.25300/MISQ/2015/39.1.10.

Thomas Eisenmann, Geoffrey Parker, and Marshall W. Van Alstyne. Platform envelopment. *Strategic Management Journal*, 32(12):1270–1285, 2011. doi: 10.1002/smj.935.

Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. Gpts are gpts: An early look at the labor market impact potential of large language models. https://arxiv.org/abs/2303.10130, 2023.

Eurostat. Use of artificial intelligence in enterprises. https://ec.europa.eu/eurostat/statistics-explained/SEPDF/cache/106920.pdf, 2025. Statistics Explained, data extracted December 2025; accessed 2026-05-07.

Annabelle Gawer and Michael A. Cusumano. Industry platforms and ecosystem innovation. *Journal of Product Innovation Management*, 31(3):417–433, 2014. doi: 10.1111/jpim.12105.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021. doi: 10.1145/3458723.

Ahmad Ghazawneh and Ola Henfridsson. Balancing platform control and external contribution in third-party development: The boundary resources model. *Information Systems Journal*, 23(2):173–192, 2013. doi: 10.1111/j.1365-2575.2012.00406.x.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2242–2251, 2019.

Ola Henfridsson, Joe Nandhakumar, Harry Scarbrough, and Nikiforos Panourgias. Recombination in the open-ended value landscape of digital innovation. *Information and Organization*, 28(2):89–100, 2018. doi: 10.1016/j.infoandorg.2018.03.001.

Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards. https://arxiv.org/abs/1805.03677, 2018.

Michael G. Jacobides, Carmelo Cennamo, and Annabelle Gawer. Towards a theory of ecosystems. *Strategic Management Journal*, 39(8):2255–2276, 2018. doi: 10.1002/smj.2904.

Cathy Jiao, Yijun Pan, Emily Xiao, Daisy Sheng, Niket Jain, Hanzhang Zhao, Ishita Dasgupta, Jiaqi W. Ma, and Chenyan Xiong. DATE-LM: Benchmarking data attribution evaluation for large language models. https://arxiv.org/abs/2507.09424, 2025.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024.

Charles I. Jones and Christopher Tonetti. Nonrivalry and the economics of data. *American Economic Review*, 110(9):2819–2858, 2020. doi: 10.1257/aer.20191330.

Paul Klemperer. Competition when consumers have switching costs: An overview with applications to industrial organization, macroeconomics, and international trade. *The Review of Economic Studies*, 62(4):515–539, 1995. doi: 10.2307/2298075.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1885–1894, 2017.

Karim R. Lakhani, Marco Iansiti, and Kerry Herman. Ge and the industrial internet. Harvard Business School Case 614-032, 2015.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 8424–8445, 2022. doi: 10.18653/v1/2022.acl-long.577.

Mark A. Lemley. How generative ai turns copyright upside down. *Columbia Science and Technology Law Review*, 25 (2), 2024. doi: 10.52214/stlr.v25i2.12761.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378 (6624):1092–1097, 2022. doi: 10.1126/science.abq1158.

Linux Foundation. Agent2agent (A2A) protocol specification. https://a2a-protocol.org/latest/specification/, 2026. Version 1.0.0. Accessed 2026-05-07.

Shayne Longpre, Robert Mahari, Anthony Chen, Naana Obeng-Marnu, Damien Sileo, William Brannon, Niklas Muennighoff, Nathan Khazam, Jad Kabbara, Kartik Perisetla, Xinyi Wu, Enrico Shippole, Kurt Bollacker, Tongshuang Wu, Luis Villa, Alex Pentland, and Sara Hooker. A large-scale audit of dataset licensing and attribution in ai. *Nature Machine Intelligence*, 6:975–987, 2024. doi: 10.1038/s42256-024-00878-8.

Francisco J. Mata, William L. Fuerst, and Jay B. Barney. Information technology and sustained competitive advantage: A resource-based analysis. *MIS Quarterly*, 19(4):487–505, 1995. doi: 10.2307/249630.

Andrew McAfee and Erik Brynjolfsson. Investing in the it that makes a competitive difference. *Harvard Business Review*, 86(7):98–107, 2008.

Nigel Melville, Kenneth Kraemer, and Vijay Gurbaxani. Review: Information technology and organizational performance: An integrative model of it business value. *MIS Quarterly*, 28(2):283–322, 2004. doi: 10.2307/25148636.

METR. Measuring the impact of early-2025 ai on experienced open-source developer productivity. https://arxiv.org/abs/2507.09089, 2025.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 220–229, 2019. doi: 10.1145/3287560.3287596.

Model Context Protocol. Introduction. https://modelcontextprotocol.io/docs/getting-started/intro, 2026. Accessed 2026-05-07.

Satish Nambisan, Kalle Lyytinen, Ann Majchrzak, and Michael Song. Digital innovation management: Reinventing innovation management research in a digital world. *MIS Quarterly*, 41(1):223–238, 2017. doi: 10.25300/MISQ/2017/41:1.03.

Shakked Noy and Whitney Zhang. Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654):187–192, 2023. doi: 10.1126/science.adh2586.

Hamidah Oderinwale and Anna Kazlauskas. The economics of ai training data: A research agenda. https://arxiv.org/abs/2510.24990, 2025.

OECD. AI use by individuals surges across the OECD as adoption by firms continues to expand. https://www.oecd.org/en/about/news/announcements/2026/01/ai-use-by-individuals-surges-across-the-oecd-as-adoption-by-firms-continues-to-expand.html, 2026. News announcement, 28 January 2026. Accessed 2026-05-07.

OpenAI. Buy it in chatgpt. https://openai.com/index/buy-it-in-chatgpt/, 2025. Accessed 2026-05-07.

OpenAI. Apps sdk. https://developers.openai.com/plugins, 2026a. Now published as OpenAI Plugins. Accessed 2026-05-07.

OpenAI. Why swe-bench verified no longer measures frontier coding capabilities. https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/, 2026b. Released 2026-02-23; accessed 2026-05-07.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 27074–27113, 2023.

Geoffrey Parker, Marshall Van Alstyne, and Xiaoyue Jiang. Platform ecosystems: How developers invert the firm. *MIS Quarterly*, 41(1):255–266, 2017. doi: 10.25300/MISQ/2017/41.1.13.

Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. Large language models can self-improve at web agent tasks. https://arxiv.org/abs/2405.20309, 2024. Accessed 2026-05-07.

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of github copilot's code contributions. https://arxiv.org/abs/2108.09293, 2022.

Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of ai on developer productivity: Evidence from github copilot. https://www.microsoft.com/en-us/research/publication/the-impact-of-ai-on-developer-productivity-evidence-from-github-copilot/, 2023. Accessed 2026-05-07.

Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. Do users write more insecure code with ai assistants? https://arxiv.org/abs/2211.03622, 2022.

Margaret A. Peteraf. The cornerstones of competitive advantage: A resource-based view. *Strategic Management Journal*, 14(3):179–191, 1993. doi: 10.1002/smj.4250140303.

Michael E. Porter. What is strategy? *Harvard Business Review*, 74(6):61–78, 1996.

Michael E. Porter. Strategy and the internet. *Harvard Business Review*, 79(3):62–78, 2001.

Thomas C. Powell and Anne Dent-Micallef. Information technology as competitive advantage: The role of human, business, and technology resources. *Strategic Management Journal*, 18(5):375–405, 1997.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, 2020.

Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 33–44, 2020. doi: 10.1145/3351095.3372873.

Carl Shapiro and Hal R. Varian. *Information Rules: A Strategic Guide to the Network Economy*. Harvard Business School Press, Boston, MA, 1998.

Alex Singla, Alexander Sukharevsky, Bryce Hall, Lareina Yee, and Michael Chui. The state of ai in 2025: Agents, innovation, and transformation. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai, 2025. McKinsey Global Survey; accessed 2026-05-07.

Stack Overflow. Stack overflow's 2025 developer survey reveals trust in ai at an all time low. https://stackoverflow.co/company/press/archive/stack-overflow-2025-developer-survey/, 2025. Released 2025-07-29; accessed 2026-05-07.

Stanford Institute for Human-Centered Artificial Intelligence. The 2026 ai index report. https://hai.stanford.edu/ai-index/2026-ai-index-report, 2026. Accessed 2026-05-07.

Stripe. Agentic commerce protocol specification. https://agenticcommerce.dev, 2026a. Accessed 2026-05-07.

Stripe. Agentic commerce. https://docs.stripe.com/agentic-commerce, 2026b. Accessed 2026-05-07.

Supreme Court of the United States. Google llc v. oracle america, inc., 593 u.s. 1. https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf, 2021. Opinion concerning fair use and software API declarations.

David J. Teece. Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy*, 15(6):285–305, 1986. doi: 10.1016/0048-7333(86)90027-2.

David J. Teece. Business models, business strategy and innovation. *Long Range Planning*, 43(2–3):172–194, 2010. doi: 10.1016/j.lrp.2009.07.003.

David J. Teece, Gary Pisano, and Amy Shuen. Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7):509–533, 1997.

Amrit Tiwana, Benn Konsynski, and Ashley A. Bush. Research commentary: Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics. *Information Systems Research*, 21(4):675–687, 2010. doi: 10.1287/isre.1100.0323.

U.S. Census Bureau. Business trends and outlook survey data. https://www.census.gov/data/experimental-data-products/business-trends-and-outlook-survey.html, 2026. Accessed 2026-05-07.

Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pp. 1–7, 2022. doi: 10.1145/3491101.3519665.

Govert Vroom, Abhishek Deshmane, and Isaac Sastre Boquet. The music industry in the 2020s. Technical Note IES868, IESE Business School, 2021.

Michael Wade and John Hulland. Review: The resource-based view and information systems research: Review, extension, and suggestions for future research. *MIS Quarterly*, 28(1):107–142, 2004. doi: 10.2307/25148626.

Birger Wernerfelt. A resource-based view of the firm. *Strategic Management Journal*, 5(2):171–180, 1984. doi: 10.1002/smj.4250050207.

Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016. doi: 10.1038/sdata.2016.18.

World Wide Web Consortium. PROV-O: The PROV ontology. https://www.w3.org/TR/prov-o/, 2013. W3C recommendation; accessed 2026-05-07.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. https://arxiv.org/abs/2407.01489, 2024.

John Yang, Carlos E. Jimenez, Alexander Wettig, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

Youngjin Yoo, Ola Henfridsson, and Kalle Lyytinen. Research commentary: The new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research*, 21(4):724–735, 2010. doi: 10.1287/isre.1100.0322.

Jonathan Zittrain. The generative internet. *Harvard Law Review*, 119(7):1974–2040, 2006.

A Prompt Excerpts and Session Methodology

We sampled local Codex session evidence. We do not reproduce full transcripts. The relevant pattern is broad product request, decomposition into implementation plan, iterative feature work, screenshot-driven quality control, and defect repair. Sensitive tokens, credentials, private source URLs, and unsafe operational details are excluded.

A.1 AI Coding Environment

The paper-production environment reported by the author was GPT-5.5 Extra High in Codex Version 26.429.61741 (2429). The Myflix session metadata also records Codex Desktop sessions and CLI version 0.128.0-alpha.1. We treat these values as instrumentation metadata for the case study. They are not an independently varied treatment and do not support causal claims about model-to-model differences.

A.2 Session Trajectory

The sampled prompts show the following trajectory:

- 2026-05-02: The initial user request asked for an application similar to a premium streaming product, with a polished interface, highlighted banners, shows, navigation through content, offline downloads, storage/NAS settings, and settings controls.
- 2026-05-02: A later plan request decomposed the work into a browser-first personal media app with web, API, worker, domain, media, provider, and UI workspaces.
- 2026-05-03: Iteration focused on player controls, hover previews, timeline jumping, sparse libraries, and enriching the app with metadata examples.
- 2026-05-04: Iteration expanded TV Mode into a dedicated scheduled-playback surface with a timeline progression, program previews, and follow-on context.
- 2026-05-05: Iteration focused on Series and Movies navbar filters, duplicate genre rails, catalog repetition, improved discovery from metadata providers, and subtitle capability with sticky audio and subtitle preferences.
- 2026-05-06: Iteration focused on offline Series behavior, retained episode visibility, offline-only rails, and disabling unavailable controls while offline.
- 2026-05-07: Iteration included detail navigation, Picture-in-Picture handoff, large-season performance, and episode-row layout refinement.

We treat the sequence as scientifically relevant because it demonstrates that prompt quality and feedback loops are part of the production function. We do not present the case as a single prompt producing a finished product. We present it as an extended interaction where the human supplied goals, acceptance criteria, screenshots, and defect reports, and the AI agent translated those into repository changes and checks.

A.3 Representative Prompt Examples

The following examples are shortened and sanitized from the local May 2026 Codex session logs. Ellipses mark omissions. We preserve the functional intent of each prompt while excluding credentials, private URLs, unsafe source details, and image payloads.

1. **Category specification, 2026-05-02.**

   > "I want to create an application similar to Netflix, with ... a nice UI, having the ability to click through the different highlighted banners and shows ... download them to watch later ... videos stored on the local drive ... configure it to store to some NAS ... an offline view feature ... binge-watch things ... skipping functionalities for intros ... multiple audio tracks ... subtitles ... log in with my Trakt account."

2. **Architecture decomposition, 2026-05-02.**

   > "Build a browser-first TypeScript/PWA streaming app with a Netflix-inspired interface, Trakt as the primary collection/watch-state source, local/manual media as the first playable source, and both server/NAS plus device offline caching once playback exists."

3. **Guardrails and visual QA, 2026-05-02.**

   > "Add a guardrails-first development layer around the current TypeScript monorepo ... GitHub Actions PR checks, local verification scripts, Docker test tiers, AI-agent rules, Playwright UI/visual checks, and complete project docs."

4. **Long-horizon implementation plan, 2026-05-02.**

   > "Build Myflix ... into a full self-hosted Docker production app for household streaming: Trakt-led metadata, rich Netflix-inspired browsing, local/NAS media import, real playback, offline caching, binge watching, TV Mode, plugin-based lawful sources, and exhaustive CI/agentic verification."

5. **Account and profile expansion, 2026-05-02.**

   "I want to have a fancy start page with a login ... allow registration ... put everything in the database ... up to ... five profiles ... favorites, the listings, the things that we have there basically would be profile-specific."

6. **Screenshot-driven UI repair, 2026-05-03.**

   "I started clicking around and testing the application, and I found some bugs ... the buttons all seem to have an issue. They are not nicely centered vertically in the middle ... Please analyze the full implementation and fix that."

7. **Player interaction refinement, 2026-05-03.**

   "When you hover with the mouse over the play bar ... it gives you a small preview ... press the left or right arrow so that we jump 15 seconds ... press anywhere on the screen and it would just play and pause."

8. **TV Mode product expansion, 2026-05-04.**

   "TV mode should be a special mode ... a dedicated page ... more like a TV schedule ... a timeline progression ... preview plus text plus how long it takes and what comes after that."

9. **Subtitle capability and preference persistence, 2026-05-05.**

   "Can you please look up some sources or additional information online for libraries or APIs ... download subtitles for a given episode or series automatically ... respect the preferred user profile language subtitle setting ... audio and subtitle settings are sticky."

10. **Offline-mode defect correction, 2026-05-06.**

    "The offline mode of the Series tab shows simply no entries although I have downloaded series ... show this only in offline mode ... verify that in offline mode none of the download or update functionalities are triggered."

11. **Paper-generation request, 2026-05-07.**

    "I now want to write a scientific paper about the topic of 'Commoditization of Software' ... first research scientific sources online ... methodology based on the approach with Codex and code base, experimental evidence like prompt examples ... and qualitative results as in the screenshots folder."

A.4 Prompt Coding Categories

We coded the sampled prompts into five categories:

1. Category specification: prompts that define a recognizable software category and desired user experience.
2. Architecture decomposition: prompts that ask for a multi-layer implementation plan or structured work breakdown.
3. Feature expansion: prompts that add new product capabilities such as offline behavior, provider controls, or catalog organization.
4. Defect correction: prompts that identify broken or awkward behavior and request a fix.
5. Quality refinement: prompts based on screenshots, layout observations, responsiveness, or interaction polish.

Our coding shows where AI assistance enters the development process. The category prompt makes the target legible. Architecture decomposition turns the category into modules. Feature expansion increases scope. Defect correction and quality refinement convert visual or runtime feedback into incremental implementation. We expect commoditization pressure to be strongest when all five categories are available because a rival can move from visible goal to working substitute through repeated correction.

B Codebase Architecture Evidence

The appendix records architecture evidence used to interpret the case. The goal is not to treat code volume as scientific evidence by itself, but to show where the product separates visible implementation from controlled capability.

Table 2 supports the key analytic distinction between generic implementation and controlled capability. Generic implementation includes the visible browsing shell, rails, detail layouts, and common buttons. Controlled capability includes source provenance, provider configuration, local file handling, profile-specific state, playback/transcoding, offline retention, and validation routines.

Table 2: Myflix workspace architecture evidence. The table identifies each workspace used in the case study and explains how it supports the distinction between visible implementation and controlled capability.

Workspace	Case-study relevance
`apps/web`	Human-facing UI: browsing, detail, watch, settings, downloads, TV Mode.
`apps/api`	Controlled server boundary: persistence, auth, media, providers, jobs.
`apps/worker`	Background processing for import, offline, and provider tasks.
`packages/domain`	Shared typed contracts and domain models.
`packages/media`	Playback queues, stream selection, import, offline, intro/binge logic.
`packages/providers`	Provider descriptors and clients behind explicit boundaries.
`packages/ui`	Shared UI package.

Table 3: Architecture-to-theory mapping for the Myflix case. The table links concrete repository and product boundaries to implementation assets, service boundaries, trust/data assets, rights/provenance assets, operational capability, and dynamic capability.

Architecture evidence	Theoretical construct	Interpretation
Next.js browsing and detail surfaces	Implementation asset	Visible workflow is comparatively easy to specify and reproduce.
Fastify API routes	Service boundary	Server-side control can mediate data, permissions, and operations.
Profile/account state	Trust and data	User-specific history increases switching cost and personalization value.
Provider descriptor package	Rights and provenance	Provider boundaries encode what sources are allowed and how they are tested.
Playback/transcoding logic	Operational capability	Reliability depends on media inspection and device-compatible delivery.
Offline retention logic	Trust and operational capability	Value depends on predictable access under network loss.
Validation scripts and tests	Dynamic capability	Continuous verification supports adaptation under rapid iteration.

B.1 Architecture-to-Theory Mapping

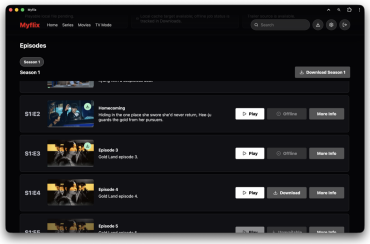
Table 3 maps the observed architecture evidence to the theoretical constructs used in the paper.

C Screenshot-Based Qualitative Observations

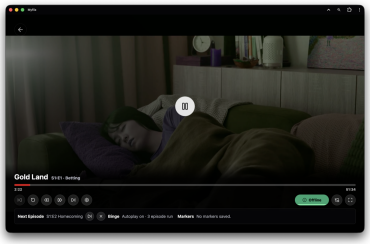
C.1 Visual Evidence Grid

We used local screenshot evidence for the screenshot analysis. Figures 7 and 8 organize the evidence grid. Cells (a)–(i) are app-facing evidence of implemented scope and visible product grammar. Cells (j)–(l) are process and repository evidence. None of the images provide evidence of legal equivalence to any commercial service.

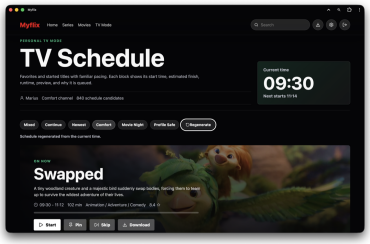
(a) Home hero, global navigation, actions, and rails.



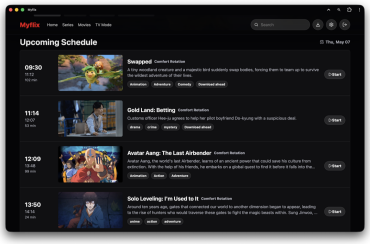
(b) Episode list with season and bulk-download controls.



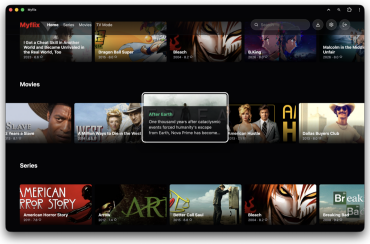
(c) Watch player with timeline, controls, and offline state.



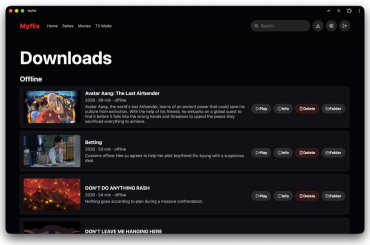
(d) TV Mode on-now card with schedule actions.



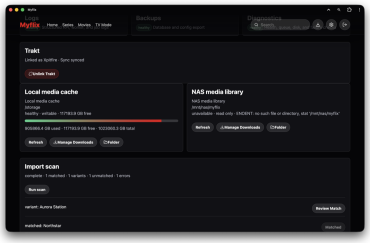
(e) TV Mode upcoming schedule with start controls.



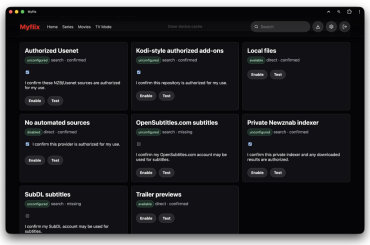
(f) Browse rails with filtered catalog organization.



(g) Downloads manager with retained offline items.

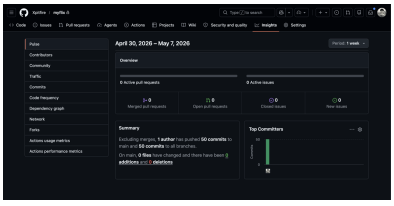


(h) Settings diagnostics, local cache, NAS, and import state.

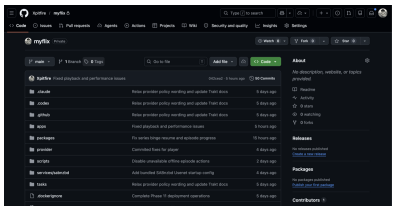


(i) Settings/provider grid with explicit source controls.

(j) Terminal validation and iteration output.



(k) GitHub Insights showing short-window commit activity.



(l) Repository file tree and recent commit activity.

Figure 8: Screenshot evidence grid, part 2. The six panels show the downloads manager, settings diagnostics, provider/source-control settings, terminal validation output, GitHub Insights commit activity, and the repository file tree. Together they document operational depth and process evidence beyond visible UI reproduction.

Table 4: Screenshot-cell observations. The table describes each cell in the two screenshot grids and explains its role as qualitative evidence for visible product grammar, workflow depth, operational capability, provenance, validation, or repository/process evidence.

Figure cell	Observation
(a) Home hero	A media-forward home surface combines a large title treatment, profile-aware navigation, primary watch action, secondary download/offline action, search, and horizontal rails. We treat it as evidence of visible reproduction of a familiar streaming-product grammar.
(b) Episode list	The series detail surface shows a season tab, episode rows, per-episode Play/Offline/More Info actions, and a bulk season download control. We interpret this as workflow depth beyond a static home mockup.
(c) Watch player	The player shows episode context, a video surface, timeline, playback controls, next-episode controls, and retained offline state. We interpret it as operational implementation behind the browsing shell.
(d) TV Mode on-now	The TV Mode surface shows current time, an on-now program card, schedule filters, regenerate control, and Start/Pin/Skip/Download actions. We treat it as recombination of catalog, profile, availability, and scheduling logic.
(e) TV Mode schedule	Upcoming schedule rows expose times, titles, artwork, metadata tags, and Start actions. The screen demonstrates generated timeline organization.
(f) Browse rails	Series and Movies navigation produces filtered genre rails and catalog organization. We classify this as a highly reproducible implementation layer when a target product grammar is visible.
(g) Downloads manager	The downloads surface lists retained offline items with artwork, summaries, and Play/Info/Delete/Folder actions. The button grammar is imitable; reliable local retention and job reconciliation are more defensible.
(h) Settings diagnostics	The diagnostics surface shows Trakt linkage, local media cache capacity, NAS library status, backup diagnostics, and import scan state. In this capture the NAS mount is unavailable with a filesystem error, and the import scan reports one matched, one unmatched, and one failed item. We read the panel as evidence of operational and configuration depth precisely because it surfaces a degraded source and a partial scan rather than hiding them. The same capture also shows a stacking defect in which status cards overlap the navigation bar, which we treat as consistent with Result 1: AI-assisted implementation reached working depth without reaching finished quality.
(i) Settings/providers	Provider cards and settings show explicit authorization, testing, and source-boundary controls. We treat this as the strongest screenshot evidence for provenance and governed service access.
(j) Terminal output	The terminal screenshot records command-line validation and iterative development output. We use it as process evidence, not as evidence of a user-facing application state.
(k) GitHub Insights	The repository insights screenshot records recent commit activity for the case window. We use it as repository-history evidence for the development process.
(l) Repository tree	The repository screenshot shows the project file tree and recent commits. We use it as repository-structure evidence.

C.2 Screenshot Descriptions

Table 4 describes the individual screenshot cells and states how each cell is used as qualitative evidence.

We use app-facing screenshots (a)–(i) to support the case-study claim that AI-assisted implementation can reproduce the grammar of a known application category. We use process and repository screenshots (j)–(l) to document iteration and local evidence. The screenshots do not establish that protected expression, trademarks, or commercial content rights were reproduced. We therefore limit the strategic interpretation to functional and architectural patterns.

C.3 Visual Evidence Boundaries

Our screenshot evidence is qualitative and interpretive. We use app-facing images to show that a recognizable media-app grammar was achieved, not to measure user satisfaction, task success, accessibility, or legal similarity. We use terminal and repository images only as process evidence. A stronger future design would recruit users, assign tasks, measure completion rates, collect perceived quality scores, and compare the result against independently built baselines. We therefore read the screenshot analysis as evidence of implemented scope, visible pattern reproduction, and development process, not as proof of market equivalence.

D Market Data and Chart Reproducibility

A reproducible descriptive-statistics layer records the 2025–2026 market evidence. The chart tables cover the following measures:

- Eurostat and OECD firm/enterprise AI adoption over time.
- Eurostat and Ireland CSO size-class adoption.
- Selected Eurostat technology types and U.S. Census BTOS business-function measures.
- Cross-source adoption measures with explicit denominator caveats.
- Stack Overflow developer adoption, trust, debugging, and agent-use measures.
- Source-backed conceptual layers for the traceable contribution economy figure.

The chart-generation workflow loads the tables with pandas, builds Vega-Altair charts, and emits vector figures, PNG previews, and Vega-Lite JSON specifications. The paper uses vector figures so chart typography and marks scale cleanly. The JSON specifications allow future reviewers to inspect chart encodings without rerunning Python.

The figures use restrained encodings: a paired line/bar figure for longitudinal adoption and size-class comparison, separated horizontal bar panels for mixed-denominator use-case evidence, horizontal bars for measurement-frame and developer-friction contrasts, and a compact value-stack diagram for the traceable contribution mechanism. The design choices intentionally avoid false precision. We do not interpolate missing years, merge incompatible denominators, calculate trend coefficients from non-harmonized sources, or plot conceptual mechanism rows as quantitative market size.

The source facts were checked against the cited source pages and summarized in the source notebook. Future replication should refresh the official statistics before regenerating the figures.

E Mathematical Details

We expand the model in the main text using the same notation: $t \in \mathbb{R}_{\geq 0}$, $G(t) \in \mathbb{R}_{\geq 0}$, $C_i(t) \in \mathbb{R}_{>0}$, $\mathcal{J} = \{I, D, R, T, N, K, S\}$, $x_j(t) \in \mathbb{R}_{\geq 0}$, and $\alpha_j(t) \in [0, 1]$ with $\sum_{j \in \mathcal{J}} \alpha_j(t) = 1$. A simple cost function is

$$C_i(t) = \frac{c_0}{1 + \beta G(t)} + \epsilon,$$

where $c_0 > 0$ is baseline implementation cost, $\beta > 0$ is the sensitivity of implementation cost to AI capability, and $\epsilon \geq 0$ is irreducible engineering cost from testing, integration, domain ambiguity, and operational validation. Then

$$\frac{\partial C_i}{\partial G} = -\frac{c_0 \beta}{(1 + \beta G)^2} < 0.$$

The irreducible term ϵ keeps the model from implying that software becomes free.

Let the incumbent's advantage be

$$A(t) = \sum_{j \in \mathcal{J}} \alpha_j(t) x_j(t),$$

with the same asset classes as the main text. Suppose baseline weights are α_{j0} and AI capability reduces α_I according to

$$\alpha_I(G) = \frac{\alpha_{I0}}{1 + \gamma G},$$

where $\alpha_{I0} \in [0, 1)$ and $\gamma > 0$. If the remaining weight is allocated proportionally across baseline complementary assets, then

$$\alpha_j(G) = \frac{(1 - \alpha_I(G))\alpha_{j0}}{1 - \alpha_{I0}}, \qquad j \in \mathcal{J} \setminus \{I\}.$$

This equation formalizes a relative shift: even if the absolute value of data or rights does not change, complementary assets occupy more of the strategic weight as implementation becomes less scarce.

For monetization, let subscription profit be

$$\Pi_s = nP_s - F - vn,$$

where $n \in \mathbb{Z}_{\geq 0}$ is subscriber count, $P_s \geq 0$ is price, $F \geq 0$ is fixed cost, and $v \geq 0$ is variable service cost per subscriber. Let agent-service profit be

$$\Pi_a = \sum_{m=1}^{M} p_m q_m + \tau X + L - F_s - v_s Q,$$

where $M \in \mathbb{N}$ is the number of service operations, $p_m \geq 0$ is unit price, $q_m \geq 0$ is operation volume, $\tau \geq 0$ is transaction take rate, $X \geq 0$ is transaction volume, $L \geq 0$ is licensing revenue, $F_s \geq 0$ is service fixed cost, $v_s \geq 0$ is variable service cost per operation, and $Q \geq 0$ is total service volume. As UI replica cost falls, n and P_s may face pressure unless the subscription includes scarce assets. Agent-service profit can remain defensible when the service controls authorization, data, rights, trust, or transactions.

For traceable contribution revenue, let $y \in \mathcal{Y}$ have gross event value $R_y \geq 0$. Let $w_{s,y} \in [0, 1]$ be the normalized attribution weight for source $s \in \mathcal{S}_y$, computed from similarity and model-influence evidence after rights filtering. Let $\eta \in [0, 1]$ be the contributor royalty share and let $\phi(g_y) \geq 0$ be the action fee for a governed endpoint. The contribution-settlement profit for a platform or application is

$$\Pi_t = \sum_{y \in \mathcal{Y}} \left(R_y - \sum_{s \in \mathcal{S}_y} \eta R_y w_{s,y} - \phi(g_y) - c_y \right),$$

where $c_y \geq 0$ includes compute, verification, logging, privacy, and payment costs. H5 requires $\Pi_t \geq 0$ while source contributors and service providers receive positive expected payouts. If attribution confidence is low or rights metadata is absent, the model sets $w_{s,y} = 0$ for payout purposes or routes the event to escrow/review.

F Replication Protocol

A future researcher can replicate or extend this case-study method through the following protocol:

1. Select a recognizable software category and define legal/ethical boundaries before development.
2. Record initial prompts, implementation plans, and acceptance criteria.
3. Use an AI coding agent with repository read/edit capability and standard validation tools.
4. Collect repository evidence at fixed intervals, including architecture boundaries, tests, validation routines, and major implementation milestones.
5. Capture screenshots of achieved workflows after major iterations.
6. Classify features into implementation, data/rights/trust/compliance, distribution/network, service-access, and traceable-contribution categories.
7. Compare the resulting evidence against hypotheses about implementation scarcity, complementary assets, service access, and attribution-based monetization.
8. Document failures, rework, legal constraints, and validation gaps.

The strongest extension would be a matched multi-case design. For example, researchers could compare AI-assisted and non-AI teams building the same bounded product category, then measure elapsed time, defect rate, perceived quality, maintainability, and how much of the final value depends on protected data or service access.

Table 5: Threats-to-validity matrix. The table lists the main limitations of the exploratory single-case design, explains how each limitation affects inference, and states how the paper bounds or mitigates the risk.

Threat	Effect on inference	Mitigation
Single case	Limits generalization	Claims framed as exploratory and hypothesis-generating.
No non-AI counterfactual	Prevents causal productivity estimate	We avoid estimating time saved.
Researcher involvement	Human judgment may explain results	Process described as AI-assisted, not autonomous.
Recognizable product category	May overstate effect for ambiguous domains	Scope limited to visible, evaluable patterns.
Legal constraints	Technical reproduction may not be lawful	IP and rights treated as core defensive assets.
Repository evidence	Code volume can mislead on quality	We use architecture and validation evidence as scope indicators, not as productivity proof.
Current platform docs	Agent interfaces may evolve	Sources dated by access date and treated as current direction.
Non-harmonized market data	Adoption levels may look contradictory across sources	Figures separate denominators and treat data as descriptive.
Survey self-selection	Developer and management surveys may overrepresent AI-interested respondents	Official enterprise statistics are used as a lower-friction counterweight.
Attribution-market immaturity	H5 may overstate near-term royalty feasibility	We frame token/action settlement as approximate, policy-bound, and future-testable.

G Future Empirical Designs

A controlled experiment could assign teams to reproduce a bounded software workflow under three conditions: no AI assistance, code-completion assistance, and agentic repository assistance. Each team would receive the same screenshots, user stories, and tests. Outcome measures would include elapsed time, passed tests, maintainability score, accessibility score, security review findings, and subjective product quality. A second phase would ask independent users or buyers whether the result is an acceptable substitute. That design would separate developer productivity from economic substitutability.

A field study could examine startups or internal-tool teams using AI over several months. Researchers could measure which assets become bottlenecks after implementation accelerates. Possible bottlenecks include data access, legal approval, security review, integration credentials, change management, and customer trust. That design would test H2 more directly than the present single case.

A platform study could examine agent-readable service adoption. Researchers could compare firms that expose controlled tools through MCP-style servers or app SDKs with firms that rely only on human-facing web UIs. Metrics could include agent referral volume, transaction conversion, integration reuse, support burden, and revenue mix. That design would test H4.

A traceable-contribution study could use a bounded corpus of licensed documents, code snippets, and service endpoints with known provenance. Researchers could generate outputs under controlled prompts, compute similarity and influence scores, compare attribution against known source use, and simulate payment splits under different thresholds. Outcome measures would include attribution precision, false-positive payouts, false-negative contributor exclusion, transaction cost, privacy leakage, and contributor acceptance. That design would test H5 without claiming exact token-origin tracing.

H Threats-to-Validity Matrix

Table 5 summarizes the principal threats to inference and the mitigation used in the paper.

Table 6: Evidence checklist for major claim types. The table links paper claims to cited research, market statistics, local Myflix evidence, or explicit methodological limitations.

Claim type	Evidence used
AI can reduce implementation cost in some tasks	Peng et al.; SWE-bench; SWE-agent; Agentless; Myflix case process.
AI productivity is not universally positive	METR 2025 RCT and observed need for iterative repair.
Implementation-only moats weaken	RBV logic plus Myflix visible UI reproduction.
Complementary assets become more important	Barney; Teece et al.; Bharadwaj; Wade and Hulland; Myflix provider/data/trust boundaries.
IP and provenance matter	Lemley; Google v. Oracle; Data Provenance Initiative.
Agent service layers create monetization surfaces	MCP docs; OpenAI Apps SDK; Stripe agentic commerce.
Traceable contribution markets are plausible but approximate	Influence functions; TracIn; TRAK; DATE-LM; Data Shapley; PROV-O; C2PA; x402; A2A.
Recent AI diffusion is fast but uneven	Eurostat; OECD; U.S. Census BTOS; Ireland CSO; McKinsey; Stack Overflow.
Developer adoption includes friction	Stack Overflow 2025 survey; OpenAI 2026 SWE-bench Verified critique; METR 2025 RCT.
Exploratory case scope	Single-case design, non-randomized history, no counterfactual.

I Evidence Checklist

Table 6 maps each major claim type to the evidence base used to support or bound it.

Scientific quality control requires major claims to be cited, derived from local evidence, mathematically modeled, or explicitly labeled as interpretation.

J Service-Layer Monetization Examples

The service-layer thesis can be made concrete through examples. In a media application, a defensible agent-accessible service might answer: "Which authorized items can this user play on this device right now?" The answer requires entitlement, local availability, profile preferences, device capability, and playback policy. The UI that displays the result is easy to reproduce; the authoritative answer is harder.

In procurement software, an agent-accessible service might answer: "Can this purchase be approved under policy, routed to the right manager, and paid through an authorized rail?" The workflow screen is reproducible; the value lies in policy, identity, auditability, vendor data, and payment execution. In healthcare administration, a service might answer whether a claim has required evidence and can be submitted under payer rules. Again, the value lies in trusted data and regulated execution.

We use these examples to show why monetization can shift from access to action and attribution. A firm can charge for the reliable completion of a governed action, for a transaction, for an outcome, for a verified data response, for licensed access to an authoritative service, or for attributable contribution to a generated response. We distinguish that basis of value from charging only for a human-facing screen that an AI system can help recreate.