Vibin Labs

Get Your Stack Back!!!

Milton Friedman, The Grateful Dead and The Unbundling of Enterprise Software

Part II: Narrower Than AI Boosters Claim and More Real Than Skeptics Realize

Robb Gaynor

April 2026

Executive Summary

Businesses are going to start building some if not all of their own software using a software factory model of production. This will change the way software is sold. Enterprise-class software vendors will also change to meet these new demands of the marketplace. All of this is brought about by the dramatic drop in the cost of software production. Businesses are gonna get their stack back!!!

A conversion to the software factory model is driven by two (2) primary behaviors:

1. Cost Savings. The factory has a huge economic benefit for those companies who convert. This is a big driver for many companies.
2. Niche Apps & Efficiency. The factory can crank out software specific to each business's needs. The business will develop apps to solve challenges throughout their operation, seeking efficiency gains and also getting the economic benefits of the factory model.

In this future of software abundance software providers will move from selling software to selling software blueprints and software operations.

Software just got a whole lot cheaper to produce! See the study, 1 person, 33 apps, 75 days, 405K lines of code. Cheap, abundant software. Regular companies and normal people are now able to produce and build complex, working applications using a software factory. The term coined last year was "Vibe Coding" and a software factory is enterprise-grade vibe coding on steroids.

When the marginal costs of software production move towards zero, the whole game changes and monetization will move from selling software to operating software.

In this paper, part 2 of a 3-paper series, we explore what happens when software is now so inexpensive to produce. The simple answer is costs and pricing will drop. And the services surrounding the current SaaS model will unbundle as someone still needs to operate the software that a business or individual builds. The amount of software will explode as regular people and many small and large companies will choose the economic advantages of the software factory over the traditional legacy model.

This is a big change. Cost of production drops. Price drops. Hard to sell at a premium what is now an abundant resource. As prices compress, services unbundle and all that's left to sell is operations. Keeping the lights on and charging for Ops is the future as how can you charge for software when it is so cheap to produce?

In the new model, software operations will become a huge deal. If lots of companies are building apps, who operates all that software and where does it run? Software Foremen will emerge to keep the lights on. Today the SaaS model provides this as part of the bundle, in the future operations becomes the core value proposition as we see an unbundling of the value proposition. Data center, maintenance, ongoing 24x7 support, those will be the valuable things a company will pay for, not the code.

And to accompany the software factory and operations model will be blueprints, software blueprints. The source code. And it's free, or virtually free to produce. Companies will appreciate having blueprints to follow when they embark on a project. A software blueprint encapsulates domain expertise. The blueprint is the requirements baked into source code. A blueprint gives a company a leg-up, a place to start when they convert to a factory. Software blueprints, like the software itself, will become abundant. Public patterns for all to follow.

Long ago, we created the ASP/SaaS model. The original driver for the SaaS/ASP bundle was in fact spreading the rather high costs to many customers for a very expensive resource. Now, those costs have evaporated, and a big driver for moving to an ASP multi-tenant architecture is gone, people can have their own stack back!

The great news is software factory applications are built to operate; purpose-built test automation, production monitoring and app quality assurance, all built in from the ground up. An AI-driven software factory has proven to be very good at building apps, and now we have an operating model to support the true future abundance of software. Consider performing a software factory audit of your app. Are you ready to convert to a factory?

Author's Note: The Grateful Dead pioneered free music; there are parallels and lessons in their journey as they monetized the experience not the music itself. We are shifting to a similar place where the software is going to be hard to monetize when it is so cheap to produce. And Milton Friedman and the Economists… No, that was not a band you saw at the Stone… The economists tell us, marginal cost of production dropping, etc. It's a real thing, a real fact of basic economics, not necessarily Dr. Friedman's specific theory. We should observe this principle playing out once again as we witness software costs drop so dramatically. So, we learn from tapers of the Dead and our very dry, sometimes boring Econ 101 class as we enter the software reformation.

A Historical Side Note - ASP & SaaS Cost Legacy

Application Service Providers (ASP) were what we called SaaS in the early days. I worked at one of the largest original ASPs, Digital Insight. 1800 banks and credit unions 7 million end users all on a single infrastructure.

We did an ASP architecture out of necessity. There was no way to deliver all that software at a price point the thousands of smaller financial institutions could afford given the old deployment model; premise-based. So, we bundled. We put the software in a data center, we made it easy for everyone to share the infrastructure, and we called it Multi-Tenant architecture. And it worked. Any size financial institution could afford a digital platform. This model holds today.

It was a compromise. With an ASP, we compromised. And the customers compromised. Banks lost the ability to look different. We tried. It was hard. And no one could customize anything. And everyone waited for updates. Compromise. But, it was cost effective. And it worked. At Digital Insight, we had an enhancement list over

600 items long… Tough to keep all those banks and credit unions happy. But again, it was cost effective. Hundreds of people in development could be spread across the P&Ls of many banks.

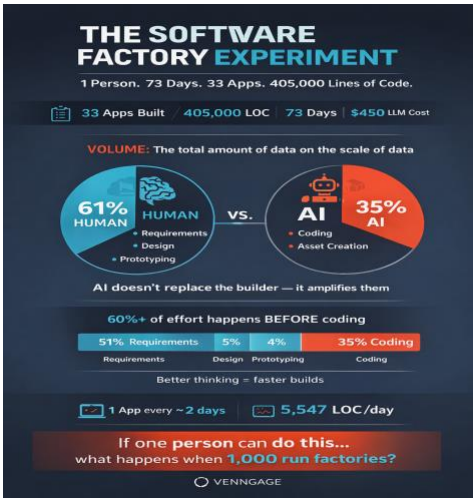
Ah, remember the days, in the original, original model, enterprise software used to be based on licensing and customization, not bundled hosting. Everyone bought their software for a license fee and ran it themselves. SaaS and ASP was a reality of software production economics as the software moved from early entrants like Chase, who ran Internet Banking in-house, to an ASP model pioneered by DI. ASP was not the preferred choice, at least not at Digital Insight.

And today. If software is cheap to produce the cost advantages of the ASP model disappear. And companies can own their own software once again. Everyone can have their own stack back. They can customize on their schedule, heck, they can customize period!!! The need for ASP is gone, i.e. high software production and maintenance costs. Everyone gets to own software again.

Software Production Costs & Economics

The unbundling of the enterprise model is enabled by several changes to the market including:

1. Abundant Software at Low Cost. The new software factory tools are producing results. A previous study on software abundance demonstrated one example of 33 apps created at very low costs and in very short timeframes. One person built a total of 33 apps in 75 days. By any measure, an example of software abundance.



2. Increased Competition. Given these new circumstances, software competition will proliferate.
   - Build Your Own. Some companies with high domain expertise will build their own solutions, they will be able to recreate core applications that they have

bought or built and do it in a fraction of the time and a fraction of the replacement costs and 100% bespoke to their requirements.
    - New competitors. Some new vendors will enter the market in specific verticals. These new entrants will have radically different cost structures based on their usage of a software factory.
3. Competitive Unbundling. New competitors with very low software production costs will unbundle pricing and move value from the software to operations where there is still value to provide, to run the software.

Incumbent vendors will be impacted on both sides of the funnel: they will see the market reprice and they will see a potential loss of customers who decide to build their own. The market forces will bring about change for everyone, and software will become abundant.

The New Model – Unbundle, Own The Code & Pay For Ops

Bottom line. The new model is focused on charging for operating software not building it. Everyone will be able to build software; the marginal costs will decline over time. What's left if you can't charge for the software as so many people have access to a cheap means of production? Software operations emerges as a key discipline and value proposition.

Today, there are five (5) primary components to the current SaaS economic model:

- Software. Initial software and maintenance, pay for the software on a subscription basis.

- Software R&D. Upgrades and updates are included in the software charges outlined above.

- Maintenance. A maintenance charge can be added as a percentage of annual charges OR at times it is bundled in the total monthly costs.

- Data Center. Receive the data center as part of the bundle of services.

- Operations. Receive operational support to keep the software running as part of the bundle, includes DevOps, 1st tier support and reporting.

A Word On Implementation Services & Costs

Implementation services and costs are not considered in the above model as they tend to be factored in as "bad" revenue when compared to recurring revenue. And implementation costs are consistent in both models, old and new.

In fact, when software production is cheaper and faster, implementations and rolling out software remains the largest and most expensive task. This does not change with a software factory.

To address this, the software factory comes with a set of tools. An education and training platform always accompanies a good factory product. Testing websites to organize the process and collect results can also help. And the software itself is designed for ease of use and can be changed at will to meet the emerging needs of the users.

So yes, some parts of the process can be further automated when rolling out software, but it remains a time-intensive and challenging task. Getting people to use software has always been and will continue to be challenging.

Legacy SaaS Model Cost Allocation

How much cost and expense can we really cut out of the model? How inexpensive will products get? How much does it cost to operate software, and will those costs become unmanageable over time when everyone has a custom solution?

All excellent questions!

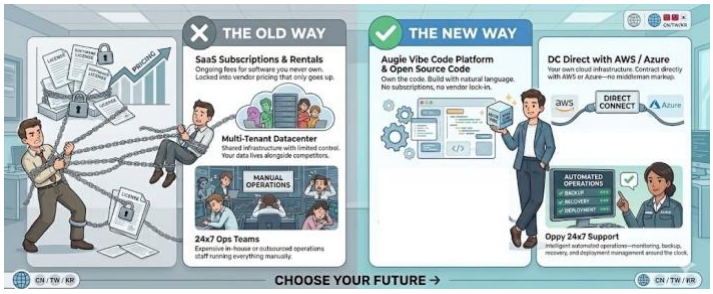
The simple answer is, the same, it will cost the same. It will be the same support burden, and no, custom dedicated stacks can be managed at scale.

We have years of experience running the SaaS model. At my previous SaaS company, the split was 65% software 35% operations. Our biggest cost was people and engineers were a majority of resources, not a vast majority, but a majority ☺. Software factory operations will approximate the costs of historically maintaining a SaaS model.

Maintenance is also simplified. Software factory platforms are built to maintain. Monitoring and maintenance are in many ways automated vs a legacy platform. Regression testing is 100% coverage and 100% automated. Every small change gets thoroughly tested. Again, things that are hard on a legacy stack. The factory Foreman will be in charge and will have automation and instrumentation at their disposal.

The percentages can be debated. Some may say maintenance and operations is 65% and invert the model. In any model at any slip in percentages there is a single emerging

fact… The reduction in software production costs cannot be debated. Software monetization will need to shift, possibly moving to the operational model, both operating the software and for some companies, running the whole show but at highly reduced costs…

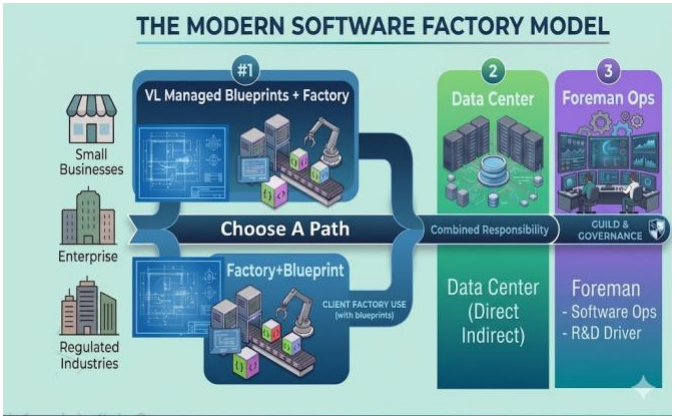


The Software Factory Model – 2 Ways To Buy

In the new model, we see an unbundling of the components. There will be two (2) primary models for the software factory implementation:

1. Companies Adopt The Software Factory Model

Many companies will want to own and develop their own software. They will use a software factory to build apps. These businesses will need Foreman operators to keep the software running. A company building their own products and outsourcing Ops, could potentially save 70%-80% of their current vendor costs once converted to the factory.



2. Companies Want Traditional Bundle with Software Factory Economics

Some companies will not want to develop their own apps. Yet, they will want to partake in the benefits of custom software, the positive economics of a software factory and avoid legacy vendor lock-in. These companies will outsource the R&D function as well

as operations and will pay an additional 20%-30% for this service. A 3rd party domain expert will manage their R&D process. Overall savings will still be 55% - 65% of traditional SaaS vendor costs.

In both new models there are fundamental differences to the way a contract is structured.

- **The Company Owns The Code**. The software component is owned by the company. They do not pay for it. They get the source code. The actual software is now the cheapest part of the equation. In some instances, the company will pay for an enterprise-grade software factory toolkit used to build new apps and maintain the code. The source code is theirs and it is free. In many instances the company is going to use an industry blueprint, the actual source code is already provided to them, they take the blueprint and start building from there.

- **Tech Partner Manages Operations**. Ops is the new revenue opportunity in the unbundled enterprise model. The software will be maintained and operated by a Software Factory Foreman for a cost, as highlighted above, the ops costs will vary SaaS vendor by SaaS vendor. This is a technical role of keeping the software running, usually in a data center environment but sometimes locally within a company's infrastructure. This is the critical new piece of the puzzle for the new model and answers the question "Who Keeps the lights on?". And software factory produced software is purpose made to be monitored and automatically tested. Software operations is efficient and cost effective.

- **Software Blueprints Drive Quality & Time To Market.** A software blueprint is domain expertise represented as a working application. Blueprints will exist for common apps like CRM and more robust apps like digital banking. Companies can take blueprints and use them as a starting point when they build their own apps with a software factory. For companies who choose a fully managed service, they use blueprints too. Blueprints also enforce a level of basic quality and compliance as all factory apps are exposed to stringent quality auditing frameworks.

- **Software Maintenance & R&D Driver**. Some companies will not want to do their own development and maintenance and will pay their Tech Partner to manage the R&D efforts. The Tech Partner will charge an additional fee for this service. Possibly the current maintenance model of around 20% can be a proxy used for how much R&D driver services will cost.

- **Dedicated Data Center Implementation**. Every business will have their own hosted environment. The data center is provided by the ops vendor or outsourced directly to AWS or MS Azure. Data centers are a cheap commodity. Many companies already maintain a relationship with a hosting company and

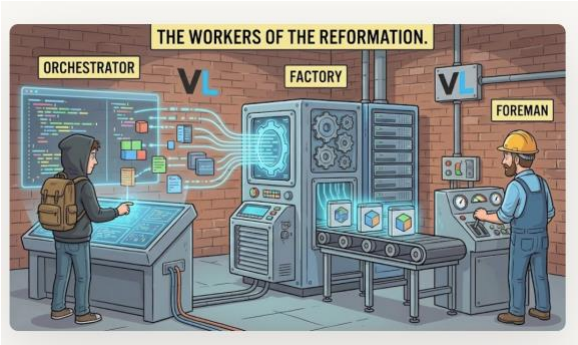
now it will be expanded to include additional software. No more ASP, the company gets full control of their technology.

Software Factory Model – How To Make It Work

The software factory operating model will be a key to unlocking this new era of enterprise software. Details of how the model works are as follows:

- Who Owns the Code. The company owns the app, it is theirs. No more sharing or ongoing subscription costs. This is a return to the older model of software ownership that dominated in the pre-SaaS era. And blueprints make this easy as a company can adopt a blueprint and have 85% of the product already completed and up to industry standards.

- Who Develops The Platform and the App? The company in most cases takes ownership of future development. Their domain experts become the orchestrators of the software factory, using an enterprise-class toolkit to build apps. In some cases, a company outsources the R&D function to their Foreman Ops partner.



- Who operates the software? The factory Foreman operates the software. This is a technical role and is focused on keeping the software running. This mirrors what SaaS vendors do today. This includes monitoring the software, managing backup and recovery and managing data center implementations when required. The Foreman sets up a service level agreement with the company. The company and the Foreman share operational risk and responsibility. These roles will be clearly defined exactly as they are today in a SaaS/ASP framework.

- Is a software factory suitable for a larger enterprise? Yes, the software factory is made for managing the software build process for a large business. Everything is consistent and standardized. A central policy server managed entitlements and local capabilities for each employee and sets development policy for who can build, what documents are required and how to gain approval for an app. The system standardizes how apps are developed and has a single app portfolio

management solution to increase enterprise-wide visibility. The factory also enforces software quality and security standards throughout the organization. Everything is logged and can be audited. Vibe coding at enterprise-grade.

- Does the software factory create secure apps? Yes. The software factory enforces early security design architecture. Security requirements are baked in. And security testing is included as well. The software factory puts all products through a process called the Application Quality Audit Protocol, an industry standard for software quality assurance. The AQAP includes security testing. All software factory apps are built to pass this standard and are audited against the standard before they launch and after every change to the product. Software factory products are secure by design.

- Will Foreman operating costs become unmanageable? Software factory products are built to operate. The infrastructure is automated and instrumented, to a much higher degree than legacy solutions. Fixes are remediated by abundant, cheap software factory economics. Software factory operating costs will be lower than legacy and will be easier and less expensive to maintain.

- Is the new application easy to operate, robust and secure? Applications built in a software factory are built to operate. The code is purpose built to make automated testing and monitoring possible. Security is built in by design. Code quality is built in from the ground up. AI-generated code follows strictly defined security protocols and requirements; this is no different than when hand-coders wrote the apps, but software factory apps are operationalized by design.

- Is my current platform and product ready for a software factory conversion? Many legacy apps have never been really studied. Every software factory product is audited against a quality assurance verification checklist. Any code base product or platform can be analyzed and audited by a software factory auditing agent to see if the solution is up to industry quality standards. This helps determine how to get the platform or product ready for a software factory approach.

- What about Companies Not Interested In Vibe Coding Their Own Apps? Not every company will want to develop their own code. For an additional ongoing fee, the Tech Partner will provide an R&D Driver service to manage the maintenance and develop new apps for those companies who need this level of support.

- How do you manage the proliferation of software and employees building apps? The software factory toolkit is enterprise class. Employees are given world-class tools to build apps. The company can monitor the proliferation of the new software and manage the process of getting new apps into production. All from a

single server-based enterprise software factory platform. The factory is 100% auditable and designed to run in highly regulated industries like banking and healthcare.

The Great Unbundling – When Will This Happen

Slowly for sure. And narrowly. The subtitle says it all. Narrower than boosters claim and more real than skeptics realize. As has been documented, AI does work wonders. But in very narrow corridors. Software proliferation is one of those narrow channels where we are seeing big change. And it does work well, skeptics should pay attention to the proof, it's in the taste of the pudding of working factory-built apps.

And it is happening all around us. A recent WSJ article (WSJ March 2026) highlighted 4 large US companies building their own solutions. Another recent study suggested that 46% of CIOs are open to replacements. The economics are too stark to ignore, and the benefits abound once a company owns its stack once again. Slowly for sure, companies will start to explore building their own software factory.

But we must not become distracted. And Agentic AI and the Agentic pundits are distracting. If any of them had worked with an LLM they would know through experience the law of necessitated determinism: out of the failure of AI, we need rules, deterministic rules to guide the AI. 100% autonomous agentic AI is a potential distraction driven by the hyperscalers to justify their dreams of token consumption.

The race for companies to harness software factories has begun. As an accompaniment to this paper, I have built a Software Abundance Calculator for companies to evaluate their internal software portfolio and find the best opportunities to start the process. Even the calculator app was built with the factory, a 10K app built by the factory for the factory. A tool generated in 6 hours for a single purpose, built for free, and used to figure out a company's next step in the reformation. Reach out, head to my website and I can get you the calculator and you can begin the journey.

It does all come back to basic economics. If cost of production falls, so does competition and subsequently, price. If price compresses, it forces unbundling, due to the single largest price component collapsing. What's left to charge for? Operations.

Welcome to the software reformation.

Narrower than boosters claim and more real than skeptics realize.