Vibin Labs

The Software Manufacturing Process

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

Robb Gaynor

April 2026

Executive Summary

Software manufacturing is now a thing! It represents a new trend; the manufacturing process applied to the creation of software.

The manufacturing sector, as defined by the U.S. government, comprises establishments engaged in the transformation of raw materials using a multiple step assembly line to create finished products in a plant or factory.

Software manufacturing follows this same model; transforming domain experts and LLM-coding (raw materials) using a standard requirements-driven process (method and assembly line) to produce software applications (finished product).

Software is now manufactured just like so many other important products, transforming from hand craft to manufacturing. It is manufactured in a factory using a requirements-driven process. The software factory is run by people and aided by the LLM-coding machine.

Manufactured software is created rapidly, at low cost, and is built to operate. The software has consistent architecture, is easy to change and customize and is capable of containing 100% automated regression testing. Operations can be fully automated and instrumented. All code is bespoke and built to make operations easy and inexpensive.

Just like the output of other types of manufacturing factories.

Given the benefits of manufacturing software, everyone who is currently hand-crafting software will have an opportunity to transform:

- Company Founders - New Vendors
- Internal IT Departments
- Legacy Hand Crafted Software companies

A word on resources. The factory is hiring. Like history shows us, manufacturing leads to lots more of the product being produced. We need many, many people to run those software factories, engineers, product managers, designers, and non-software folks alike. We are going to need a massive amount of labor to convert to the factory model…

The software manufacturing process is a direct result of my experiences building software products with a factory approach. These experiences may or may not apply to everyone else, I freely admit this. Software factories also do not solve every challenge. If the requirements are incomplete or the product is a huge, complex, distributed code base, the software manufacturing principles might fall short. And the moat of existing legacy vendors is a real barrier that the software factory has to learn how to manage. It will be fascinating to see how software manufacturing tackles these challenges…

This is the way! The more you look at it, the more obvious it becomes. The software production process can benefit from adopting the manufacturing process and some percentage of hand-crafted coding will convert to a factory model in the coming weeks, months, and years.

Software Manufacturing – The Raw Materials & The Method

Software manufacturing can now produce complex, working production applications. There are numerous, well-documented proof points documenting the output of a software factory (link to paper 1). Manufacturing software works at scale and can manage complexity and implement with sophistication.

The software factory follows the same pattern as traditional manufacturing systems and processes. Transform raw materials into finished goods.

The Orchestrator & Builder - The Human Domain Expert

A person with deep, intimate knowledge of a specific area or domain is the primary app builder. They orchestrate the entire manufacturing process. They validate constantly as the LLM needs to be proactively managed. A domain expert can turn their knowledge into an application. The domain expert drives the research, specifications definition and design, which constitutes over 70% of the overall software manufacturing process.

The LLM Coding Machine

Any LLM can be used to create and compile the code in the software factory. The LLM follows the strict specifications and protocols of the factory and builds the app in the preferred language. The LLM can compile and write code in any software language. The LLM is monitored and verified constantly throughout the process. LLM-coding equates to 30% of the manufacturing process.

The Methodology & Process

The domain expert and LLM cannot produce software without the third component, the software manufacturing methodology and process. Software manufacturing methodology relies on a requirements-driven process and stringent verification at multiple stages. The methodology is the critical third component acting as the labor in the sense that it does much of the work to shape the final product. The method is the organizing principle for the factory manufacturing process.

The Outcome – Manufactured Software Benefits

1. Fast Production Times & Low Cost

Software manufacturing builds apps rapidly and at low cost. The largest cost is the domain expert orchestrator. The LLM costs are negligible as the software manufacturing process uses tokens very sparingly. And time to market is measured in hours and days not months and quarters. By several measures, manufacturing time to market is achieved faster than in a hand-coded effort.

2. Easy To Customize

Companies and people get their stack back with dedicated infrastructure, no more sharing of bloated SaaS infrastructures. That's right, the bloat is not a moat!!! Factory-manufactured code is built to make it easy to change and modify the app. This all leads

to huge flexibility and allows the app builder to go at their own preferred pace. Roadmaps are reduced to nothing, as domain expertise becomes the new bottleneck.

3. High-Quality Standards

The factory has high-quality standards enforced at multiple stages in the process. Verification is a cornerstone of the factory method. Quality standards are built into the requirements and specifications. The software is audited continuously against stringent industry quality guidelines. Lastly all manufactured software has built in coverage for automated regression testing. Every change is tested in minutes across the stack. This testing approach is an outcome of the factory and can be hard to replicate with hand-crafted applications.

4. Ease Of Operations

Manufactured software is built to operate. Testing is automated and deployment is carefully managed. All elements of the architecture are proactively monitored and instrumented into control panels and operations dashboards. Backup and recovery and 24x7 support are standard outputs of the factory.

5. Standardized Architecture

Manufactured software is developed using a standard, well-documented process that enforces consistency. Everyone follows the same process. The architecture is built to optimize the LLM-coding machine's capabilities and modularized for scale and ease of support.

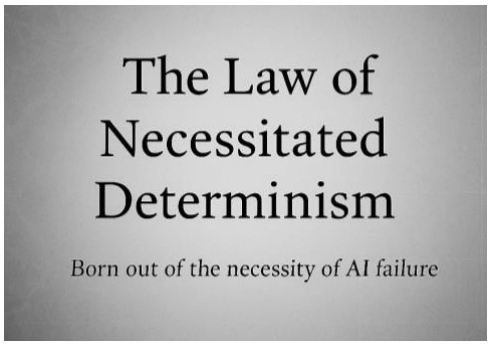
6. Documented & Auditable

All software factory output is fully documented and auditable. In highly regulated industries, the software factory creates a very high level of transparency for all stakeholders and regulators. Everything is tracked and every activity can be reported on. Software manufacturing lends itself to highly regulated environments that demand repeatable structure.

7. Efficient LLM Usage

Software manufacturing uses the LLM efficiently. The emphasis on upfront thinking and standard factory protocols allows the LLM to compile the spec and build an app using minimal LLM metered tokens. Iterations are minimized and surgical editing allows for the LLM to become an efficient factory machine-partner. As LLM costs fluctuate, the factory can provide a buffer as efficiency is an outcome of the software manufacturing process.

The legitimate questions about manufactured apps must be answered. Is it secure, who will operate, is it scalable? All great questions. The world will learn in the coming years as we launch into the transformation to software factories. And fortunately, the news is good. Manufactured software is secure, built for quality, built to operate, and built to last. There is a reasonable future where manufacturing techniques help lead to an abundance of software and an abundance of people orchestrating and running software factories.



The Law of Necessitated Determinism – Born Out Of The Failure Of GenAI

The software manufacturing process relies on an LLM to provide the coding and compiling engine. LLMs regularly make mistakes and it is no different when they write code.

The Law of Necessitated Determinism is born out of the necessity of AI failure. AI fails and we have to be deterministic in response. It's simple. I wish it were different. I wish AI were autonomous. It is not. At least not today.

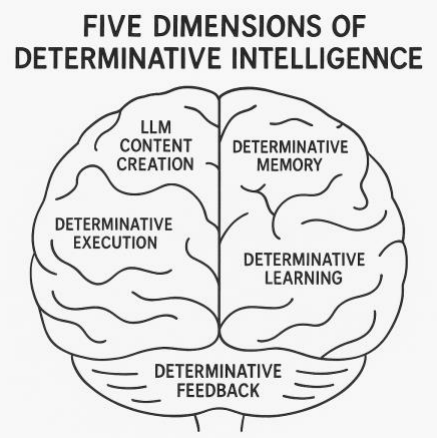
What is the counter to the LLM wandering? Verification. Trust and verify. Everything. At every step. Over and over again. In a multi-step process, the LLM wanders, forgets things. The orchestrator must check the work at every step. It's slightly time-consuming but a necessity.

The LLM knows they will be verified and they make it easy to check their work, always accepting credit for incomplete assignments when they inevitably fail. And, there must be final verification. All software methodologies have final testing and software manufacturing is no different. The apps are regression tested and audited to make sure they meet strict quality guidelines. Verification is driven by the orchestrator domain expert and is built in at all stages.

The software manufacturing method and process are the guardrails that manage the LLM's output and tell the human when to verify. Without the methodology, there would be no factory output. The human can't do it themselves. The LLM can't do it autonomously and the human and LLM combined are still deficient. When all three are combined and use manufacturing technique, the factory becomes productive.

How The Software Factory Manages the AI LLM

The capabilities and limitations of artificial intelligence Gen AI are well understood. Below is an outline of how manufactured software manages the various LLM challenges.

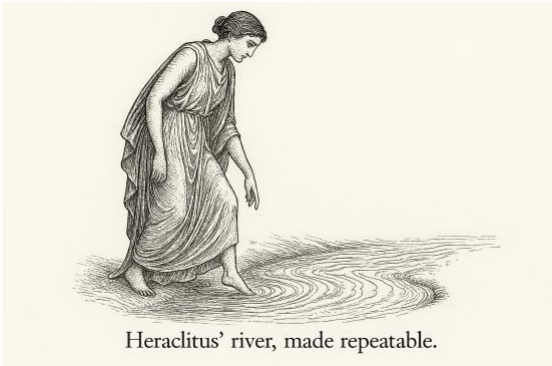


The Factory Manages AI Probabilistic Nature with Determinism

GenAI is probabilistic. LLMs are really good at guessing the next word. Really good. That capability is at the heart of how Gen AI works. The good news is the LLM likes rules. The more rules, the better the outcome. The software manufacturing process is built around this need for determinism.

Software Manufacturing Requires a Multi-Step Process

Real work requires a multi-step, at times iterative process. On their own, LLMs struggle to iterate. When an LLM follows a series of instructions, it wanders or hallucinates as the term has been coined. If all work were one-shot and done, the LLM would be perfect. The software manufacturing process coupled with the human builder enables the LLM to do iterative work and produce software.



Software Factories Need Patterns & Memory – LLMs Don't Remember

Software manufacturing inherently must remember things…. The factory is based on standardized memory; specifications through design and prototypes. LLMs do not remember. It is true, LLMs can remember bits and pieces of your conversation and context but that capability is minimal. After a few prompts, an LLM session will lose context. And don't get me started about iteration…

Keep in mind the AI companies do keep your data around for 30 days or so and they use your data to train and improve the next model. Enterprises can turn these trackers off and go Zero Data Retention (ZDR). That's not memory, that's just usage logs.

Software Manufacturing Adds Feedback for the LLM

The software factory provides feedback to the LLM-coding machine. The feedback includes learnings from previous experiences (protocols) and real-time feedback in the form of regression testing and continuous software auditing. For the LLM the feedback loop is a key component to achieving high-quality, easy-to-maintain software.

Software Manufacturing Verification – Digital Banking Case Study

Software manufacturing relies on verification. Below is a case study demonstrating how verification works when applied to an example of factory-manufactured software, in this case a digital banking platform.

- Regression Testing. Every new feature or enhancement for the banking platform is tested across 100% of the platform. A person executes an automated bundle of 26 test suites and 373 individual tests covering the entire platform. Each test output is verified, and the platform update is released by the person responsible for the new feature.

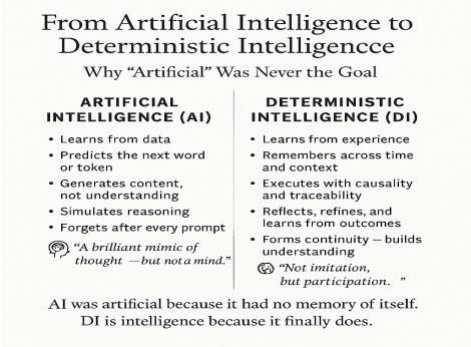
- Full Software Auditing Four Protocols (Quality-Security/Architecture/Specs/Design). The banking app is tested against four audit protocols including a quality review of 400+ points ranging from security to code quality and completeness and performance. The software audits run after every change to the platform and generally on an ongoing basis to ensure continuous compliance.

- Requirements Audits. A new banking feature development or enhancement includes two requirements audits; an audit is conducted against the LLM plan for building the new feature or app and an audit is performed after the app is developed to ensure it complies with and includes all original requirements. These audits are performed and driven by the person building the application in conjunction with an LLM. The documents become part of the build archive.

- ADA compliance. The digital banking platform is tested for full WCAG 2.1 Level AA compliance. These tests are automated as well as manual. Every new feature or enhancement is audited against the ADA standard. The automated testing minimizes the need and scope of the manual test, and both tests must be completed for each change.

The digital banking platform was built for quality and to support a continuous process of monitoring and improving quality. Augmenting the person's ability to verify is the key to a successful and robust digital banking platform.

The Software Manufacturing Future – Transformations



Software is now manufactured in a factory. Using a person who is an expert and an LLM-coding machine, the software factory produces high-quality, production-ready code. All people who built software will develop manufacturing techniques.

- **New Software Founders & Young Companies**. New entrants will use the software manufacturing process to maximum benefit. They will convert willingly as they don't care how the code gets produced and they're not tied to the economic model of software scarcity. This segment will adopt the factory rapidly.

- **Corporate Internal IT Departments**. These folks will love transforming to software manufacturing as their primary competency is not software, they had to become developers by necessity. Manufacturing software will revolutionize the internal app portfolio of all companies. The software manufacturing process fits well inside a company playing on their need for standards and auditability. And the cost and time to market benefits will be huge. They might replace incumbent vendors, they might not, time will tell.

- **Legacy Software Vendors.** Huge code bases will be no excuse for not adopting manufacturing techniques. Legacy companies will change from hand-coding to manufacturing of code. Their bloated moats of millions of

lines of code are irrelevant when software can be manufactured eventually by lots and lots of domain experts. They will attempt to transform to manufacturing models.

An Abundance Of Labor – We Are Hiring

State school education and even I can do the maths (UK context)…. There are something like 500,000 software companies worldwide over $10 million in revenue. In addition, there are over 1 million companies large enough to have lots of software running in their organizations, they are internal IT transformers. And there are 11 million additional small businesses across both categories when you go to tracking over $1 million in revenue. They all use software and they all could potentially convert to a software manufacturing model.

That is a huge amount of transformation and a huge amount of ongoing maintenance of the manufactured code base.

And abundance, don't forget about abundance. Every time we start using a factory to produce a finished good, output explodes. Not 2x or 5x but 100x (textiles) or 1000x (semiconductors). Again, that abundance points directionally towards lots of work, the software factories are hiring.

And where will these people come from? There are 40 million engineers world-wide, they might provide a partial solution. If you could retrain every engineer and turn them into a transformer of software, and if you turned the rest into the factory foreman needed to run everything, you still fall woefully short. What we are looking at is a labor shortage. We need the engineers, we need other software related folks, and we need domain experts, etc. We need 'em all.

We are talking about a lot of work to transform to the factory, good news, and we need lots of people to help, also really good news.

The Stages Of Transformation

Start small. One project. One module of your platform. One app in your portfolio. Prove out the model, learn the process. Start on a new project or a new feature. Or convert an existing module or component.

Think big. How will you transition your entire portfolio of apps you use internally? How will you rebuild your platform and transition paying customers? This will take time and will represent a program of work over months and potentially years.

Verify everything. The human orchestrator, the factory foreman, these people stick around. This is not an autonomous factory; it is a manufacturing process driven by a human. And a big part of their role is verification and quality control.

Transform. In the end, everyone has a choice. Convert to a factory where it makes sense right now or continue to craft code by hand. You have alternatives. Transformation has advantages if you buy into the manufacturing mentality. Some will stick with hand crafting. Some will transform. Manufactured software will provide some with advantages when they transform to the factory.

This will all take time. Transformations will be learning opportunities executed over a program of work. Narrower than AI boosters claim and more real than skeptics realize.

In Conclusion - The Software Reformation Has Started

The software reformation has started. Software manufacturing is going to be part of our future. The potential benefits are worth pursuing. In the future, software will become abundant and proliferate everywhere. And we need lots of people to drive the bus!

The next stage of the exciting evolution of software is here, going from hand-coded craft industry to a manufacturing industry. The scale of new apps will potentially be unprecedented. Join the software reformation and become a software manufacturer! The future of software is happening today!!!

Next Up – The Software Manufacturing Methodology

The next paper in this series will discuss the software manufacturing methodology. There are three key components to the software factory; the person and domain expert, the LLM-coding and compiling machine and the structured methodology.

The software manufacturing methodology is the SDLC of the software factory. The specific process will be dissected and discussed. Like other SDLC examples, the factory method relies on structure, an emphasis on up front thinking and stringent verification at all stages. The method ensures repeatable high-quality results.