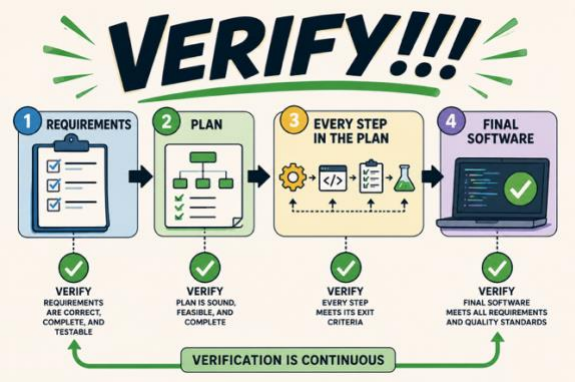
Vibin Labs



The Fintagonist

A contrarian's view on artificial intelligence and software manufacturing

Assume Breach! Trustless Verification of AI Integrity.

Austin, TX, May 9, 2026

AI can make a person really productive. This same AI also fails regularly even when it is given very explicit instructions. To harness AI, first give the AI rules and a method and then, the AI must be validated constantly as the integrity of the results varies. This is not Ronald Reagan "Trust but Verify" this is "Trustless and Verify". Mr. Reagan tried to start from a position of trust. With AI, it's best to start from a position of assuming the results contain errors and failures. Do not trust the results and act accordingly. Trustless. A good rule set, coupled with verification and human checks and balances creates real results with AI.

Assume breach. Assume failure. And verify results relentlessly… This week we are taking a break from the 4-paper series which has been the focus these past 3 weeks. This paper is a bit nerdier than usual, getting all AI techie…. I get to delve into my fav topic…. Validation & Verification.

With artificial intelligence we must assume failure and we must verify. Trustless Verification of integrity. This mirrors the real-world experience of digital banking. All along, we assumed breach. We assumed the user's credentials had been stolen. Security features such as out of band authentication and multi-factor authentication are all based on this principle. Assume breach and plan accordingly.

A great framing for this discussion was published a few days back in Forbes (Forbes May 7, 2026: The 1% Catastrophe – Why AI Agents Drift…). The article makes the point that LLMs wander, and you need to manage that characteristic. It's an interesting read and great background for the discussion of how we manage wandering AI in the software factory!

The first time AI failed me, probably also the first time I used AI, I was disappointed. The second time it failed, probably the second time I was using AI, I was discouraged. The

third time it failed, I realized I could not get very far without checking the work. In fact, I could not go more than 1-2 steps without checking.

How did I handle the failures of AI? My first attempt was to guide the LLM with rules. Lots of them. These are commonly referred to as guardrails. Instructions inside the workflow. Rules to make the LLM output better and avoid common missteps.

Building rules inside the process doesn't work. Guardrails are regularly ignored by the LLM.

OK, so audits. Let's get the LLM to check its work. So, I started auditing the LLM output. I asked the LLM to make sure it had done everything I had asked, had it satisfied the checklist, the requirements. The LLM said yes, it had. It lied. It had not. It had not even bothered to check the actual work; it just decided to tell me it had.

This was a sad day for me. The LLM could be given rules which it generally followed, then it will tell you it followed all the rules when it had not. Totally 100% untrustworthy.

The answer was Merkle hashing and Proof Bundles…. What???

Fortunately, before AI trustlessness became an issue, we invented a solution for gaining quality through external verification. In various industries, validation concepts have existed for decades and use fancy terms like Merkle Hashing and Proof Bundles. These concepts were developed to validate the quality of a work product, to validate work using an external, verifiable, auditable process.

That same concept applies to managing AI. The solution basically is validation and logging. If you validate often enough for the right things and record everything, the results eventually improve enough to where you can use the outcome. AI must be continuously verified both systematically and by a person. Both steps of verification are required.

Everyone is arguing about fully autonomous agentic AI and how to make AI agents actually work. The successful agents seem to have one thing in common. Humans verifying their work. These agents are not autonomous, far from it.

So, how does the software manufacturing factory handle verification?

It starts with a requirements driven manufacturing process and incorporates constant and rigorous verification. A working model for software manufacturing validation is outlined below. The big message. Augmentation is what drives successful AI deployments. This is not about autonomous agentic agents making it up as they go along. It's the opposite. Deterministic Software Agents, rules based and constantly verified… At VL we call them DSAs and DSAs are way into verification!!! And the result is high quality software made in a factory, produced in a fraction of the time when compared to legacy software development.

The good news is the software factory works. In a manufacturing role, the LLM driven by a person, a great set of requirements and validated constantly, creates high quality software. Without it, you get the toy prototypes built by the vibe coding tools. Verification is a key to great AI output.

Below is an outline for the factory protocol currently in production at Vibin Labs. This is meant to document best practices in software manufacturing, it is not meant to be a critique on current Vibe coding and AI-driven development tools as those tools invariably have their own answers to managing an AI successfully. Read on to gain insight into how one company is solving the Trustless Verification challenge.

----------MEMORANDUM----------

To: VL Factory Floor Orchestrators & Factory Foreman

FROM: Vibin Labs Factory Standards Committee

RE: Software Factory Verification Standards – CONFIDENTIAL

Software Verification Protocol V 2.37.996

This information is confidential and the property of Vibin Labs.

Software Manufacturing – The Orchestrator & The Factory Agents

All software manufacturing projects must have two (2) key roles as follows: a person orchestrating and validating and an LLM doing their part to verify and analyze.

1. The Person - Domain Expert Orchestrator. The person orchestrates, moves the manufacturing process through its steps and is a big part of verification. In the current Vibin Labs software manufacturing model, the human represents 61% of the overall effort split between defining specs and verifying results as documented in this VL analysis:

   (https://vibinlabs.com/SoftwareFactoryExperiment_Mar2026.pdf)

2. The AI Agent, - A Deterministic Software Agent (DSAs). An AI agent is heavily involved in verification and uses a set of rules and standards to constantly check the work. This is an LLM verifying the coding quality of a different LLM. And doing it deterministically.

3 Key Components to Verification Success

All software manufacturing projects must have these three (3) elements for a project to gain approval:

1. Requirements and Requirements Inventory

The simple answer to software factory verification is to verify against the requirements… The spec. The spec drives the entire manufacturing process.

2. Automated Regression Testing

   Project must have 100% regression testing built in. Software manufacturing relies on 100% fully automated regression testing. Every app is built with fully automated testing covering all tiers in the architecture from front end to server.

3. Automated Operations Environment.

   Every new project must have automated operations requirements and dashboard instrumentation and reporting built in. A solid verification system is supported by automated operations. Factory software ops is highly instrumented and the process is automated. The environment can be monitored and maintained with a minimum number of resources as every software factory product is built to operate.

Establishing Software Manufacturing Standards

All projects must have a verification framework:

1. Project Requirements Inventory. A detailed inventory of each discrete key feature in the product.
2. Software Quality Standards. A set of quality standards to be applied uniformly to all projects.

Explanation – Verification Criteria

What do you verify against? In the software manufacturing process, there are 400+ points of verification ranging from security to quality and testing standards (VL quality standards v 3.491).

The standards are split into 4 categories:

1. Software Quality Audit (AQAP). Is the architecture sound and working? Does the app pass basic code standards, are common LLM coding mistakes avoided? Are API endpoints authenticated? Is error handling consistent across the platform? Security and code quality are reviewed. Is the code complete? The AQAP covers the basics of app architecture, quality and security.
2. Product Domain Audit (ADAP). Is the product appropriate for the target user and industry vertical? Are basic standards met for usability and ADA compliance? Does the architecture support the required scale for the platform and is the cloud implementation appropriate? ADAP makes sure the app is built right for the target customer.

3. Requirements Audit (ARAP). Does the app meet the defined requirements? Is the code fully implemented for each feature? Do the features work in production? ARAP checks to verify the code matches the requirements.
4. Build Effort Audit (ABAP). The ABAP is an analysis of the effort to build the product. How long did it take, which developers worked on the product, when was code changed and updated and does all this activity align with product objectives and the basic requirements. ABAP tracks the actual build effort step by step.

Together these 4 audits and 400+ points of verification constitute the basis of what gets verified throughout the manufacturing process. Lots and lots of verification, over and over again…

The Software Manufacturing Verification Process

All projects must satisfy the following eight (8) steps of manufacturing verification.

It is understood that the verification steps are built into the overall time estimates for software manufacturing. All projects factor in the verification stages.

Project Management Note: As documented in the VL software factory abundance study, verification does not slow down the process. In fact, it makes the overall process quicker as there is less downstream thrash when issues are surfaced early and often. The factory method builds in verification, and it speeds up the overall process.

There are 8 different stages of verification driven by the human orchestrator and the DSA (deterministic software agent):

1. Requirements Verification. Verify requirements meet factory manufacturing standards. Every app must meet stringent requirements for product architecture, security and quality standards. These standards become requirements for the new solution, security and quality are built in and verified from the beginning.

2. LLM Project Plan Verification. The plan created by the LLM is verified several times. The plan is checked against the original requirements and the 400+ quality checks. Does the plan satisfy all the requirements? Does the plan deal with dependencies and is it executable?

3. Build Step By Step Verification. Every step in the build process is verified both against requirements, standards and by a visual check of the software once the app can be accessed by the domain expert orchestrator. This process ensures issues are caught early and do not compound through the phases of the project.

4. Final Product Requirements Audit. The final product is validated against the original requirements. Has everything been built, does the code exist and does

the app work as originally outlined. This is both an automated test and a manual test by the orchestrator.

5. Final Product General Audit. The final product is verified against the 400+ quality standards. Security and testing quality standards are highlighted to ensure a secure and streamlined operating environment. A final audit scorecard is created for the final app.

6. Final Audit Secondary Verification. Every finding in the Final General Audit is verified by a different AI LLM session. This audit is at a code level. The original audit result is verified to ensure it exists and can be fixed and remediated. A different LLM session must be used for this verification to ensure true independent results. Using a different LLM session ensures the level of separation necessary to satisfy this condition of an independent check. Even using the same model but opening a fresh session provides the level of separation needed to ensure quality results.

7. Final App Regression Testing. Full regression testing is built into software factory products. The full test suite is run on the final application several times and all aspects of the architecture are tested from front end applications to the server and database layers.

8. Post Implementation Ops Audit. A final audit against the 400+ quality standards is performed once the app has been promoted to a production environment. This test can be run at set intervals and becomes part of the general operating procedures for the platform.

Surfacing And Fixing Issues

All projects must have an “issues remediation” process defined and implemented before the project can begin. Verification must be coupled with rapid fixes to make the factory manufacturing process work.

At every stage of verification, issues can be fixed immediately using the software factory. The outcome of each stage of verification is a new version of the software addressing all known issues surfaced through the audit. The issues must all be addressed before the manufacturing process can continue.

Verify and Fix. Verify and Fix!!!

----------End VL Internal Documentation (March 13, 2026)----------

Final Result – Quality Software

Trustless verification leads to high quality software manufactured in a software factory. This is not possible autonomously. There is no magic button, manufacturing software takes orchestration and work, defining requirements and constant verification. The result is high quality software produced in a fraction of the time compared to legacy methodologies.

Humans in the loop, defining the loop, managing the loop and verifying the loop over and over again. Welcome to my world hahaha. I laugh. If it were only true that the AI machine could run itself, oh the days of glory and leisure we all would enjoy. For now, we are still needed. Sorry your job is not going away :-). For now, I trust not, and verify often!!!