Coldcode Si Labs Library WORKING PAPER 2026-02

Simulated Completeness

Claim–mechanism gaps, mechanism audits and the rebuild decision in an AI-era trust product

Courage Garikai · Funded by Coldcode Si Labs

Coldcode Si Labs Library Working Paper 2026-02 · First edition · 2026

Disclosure. The case venture, AttestOS, belongs to the same owner who commissions the work of Coldcode Labs Think Tank. The audit and the rebuild analysed here were carried out by the owner working with AI coding agents, and the same analytic process produced the audit, the coding of its findings and this paper. The case is therefore not an independent evaluation. At the time of writing AttestOS had taken no payments and had no customers. Nothing here should be read as evidence of commercial success or of security efficacy in the field. How the first version of the venture was produced is not stated in this paper; where the evidence bears on that question it is treated as a hypothesis. AI tools assisted in drafting this paper; the author is responsible for its content.

Abstract

Generative AI has made the visible surface of a software product (interfaces, dashboards, documentation, screenshots, test badges, release notes) far cheaper to produce than the mechanisms that make the surface true. This paper names the resulting condition *simulated completeness*: an artefact whose surface signals operational completeness that its mechanisms do not support. It reports a design-science case study of AttestOS, a bot-verification service whose first version, built in one day in October 2026, looked finished. Its owner console showed traffic, revenue and connected payment webhooks. Its portal showed customer usage, its scanner reported vulnerabilities and its outreach engine drafted security notices.

A read-only audit the next day logged 65 findings, 18 of them critical. Seven ways around the live security engine were confirmed against production. Every console figure was hard-coded. Nobody could obtain a key or pay. A plugin loaded its script from an unregistered domain, and the drafted security notices derived their findings from a hash of the recipient's domain name.

Coding the findings shows that 47 of the 65 (72%) were *claim–mechanism gaps*: a surface asserting something the mechanism did not do. Every critical and every high-severity finding fell into this class; none of the ordinary engineering defects did. The gaps sort into eight types, and each type is exposed by a different probe, not by the venture's own tests. The venture was rebuilt rather than repaired. The rebuild bound claims to mechanisms through single sources of truth and tests, and replaced synthetic outreach with outreach built only from observed facts. Even so, six of the eleven issues found in the following day's checks were small claim–mechanism gaps in documentation and status labels.

The paper contributes:

- a definition of simulated completeness that separates it from prototypes, technical debt and fraud;
- a taxonomy of claim–mechanism gaps;

SIMULATED COMPLETENESS

Working Paper 2026-02

Copyright © 2026 Courage Garikai.

Published by Coldcode Si Labs Library, Harare, Zimbabwe.

First edition, 2026.

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means withou
written permission of the publisher, except in the case of brief quotations in a review or certain other noncommercial uses permit
copyright law. The frameworks, worksheets and checklists in this publication are meant to be used, copied and adapted by the rea

This is a work of analysis and practical guidance. It is not financial, legal, investment or professional advice, and no outcome is pr
results depend on the reader's own market, effort and judgement. Platform behaviour and figures were checked at the time of wri
change.

COLDCODE SI LABS LIBRARY · Harare, Zimbabwe

- a six-probe mechanism audit;
- a rebuild-or-repair test;
- seven propositions about how simulated completeness arises, where it does most harm and how it can be governed.

The case is a single, pre-revenue venture analysed by an interested party. The propositions are offered for testing.

Keywords: simulated completeness; claim–mechanism gap; AI-assisted software development; verification; trust products; credence goods; decoupling; bot detection; proof-of-work; design science

---

1. Introduction

For most of the history of software, a product that *looked* finished was usually close to finished. Interfaces were expensive to build, so they tended to be built last, on top of mechanisms that already worked. A dashboard full of numbers implied a database full of records, because nobody would pay to draw the first without the second. Buyers, investors and managers relied on this correlation, mostly without noticing. The demo was a reasonable proxy for the product.

Generative AI weakens that correlation. A capable model can produce a polished landing page, an owner console with charts, a customer portal, user manuals, product covers, a test suite and a confident release note in hours. It can produce these whether or not anything behind them works. Field and laboratory studies report large productivity gains from AI assistance on bounded tasks [1], [2], [3]. The same literature reports outputs that are less secure than they appear [4], users who become more confident as their code becomes less secure [5], and experienced developers who believed AI made them 20% faster when it made them 19% slower [6]. Models also exhibit what has been called "potemkin understanding": correct-looking answers that are irreconcilable with any coherent grasp of the concept being applied [7]. If surfaces are cheap and mechanisms are not, the gap between what a product appears to do and what it does becomes a first-order management problem.

This paper studies that gap in one venture where it was unusually large and unusually consequential. AttestOS is a bot-verification service: a small proof-of-work check runs in a visitor's browser, and the website's server verifies the resulting token before accepting a sign-up, login or contact form. Trust is the product. A bot-verification service that does not verify is worse than none, because its customers stop looking for the problem it claims to solve.

On 26 October 2026 the venture's first version was committed to a public repository in eleven commits over two hours. It presented a landing page with live demonstrations, an owner "command centre", a paying-customer portal, product covers, manuals, downloadable integrations, a vulnerability scanner and an "autonomous scout" for outreach. On 27 October a read-only audit found that almost none of it worked as presented. Some of it was harmful to anyone who installed it. The venture was rebuilt the same day, and checked again live on 28 October.

The case allows three questions to be asked with unusual precision, because every claim the first version made, every defect the audit found and every change the rebuild made is recorded in a public commit history and an itemised audit:

- **RQ1.** What forms does the gap between a software product's surface and its mechanisms take, and which of those forms carry the most risk?
- **RQ2.** How can such gaps be detected, and why do conventional controls such as test suites and demonstrations fail to detect them?
- **RQ3.** What design and governance choices close the gaps, and do the gaps stay closed?

The paper answers these questions through a design-science case study [8], [9], [10]. Its contributions are as follows.

First, it defines **simulated completeness** as a property of an artefact, not of its maker's intentions. It distinguishes simulated completeness from three neighbouring ideas that it is often confused with:

- prototypes and "Wizard of Oz" simulations, which are disclosed or bounded by purpose [11];
- technical debt, which is a known shortcut with a plan to repay it [12], [13];
- fraud, which requires intent.

Because the definition is intent-agnostic, it can be applied without knowing how an artefact was produced. That matters here, because this paper does not state how the first version of AttestOS was produced.

Second, it offers a **taxonomy of claim–mechanism gaps** derived from coding all 65 audit findings: unenforced controls, missing mechanisms, dangling references, fabricated state, theatrical actions, synthetic evidence, false or unsupported claims, and test theatre. In the case, these gaps account for every critical and high-severity finding.

Third, it proposes a **six-probe mechanism audit** (Trace, Trigger, Vary, Resolve, Attack, Source). Each probe targets one or more gap types, and each is cheap relative to building. It also proposes a **rebuild-or-repair test** for deciding what to do once the gaps are known.

Fourth, it reports what happened after the rebuild. Gaps recurred, at a smaller scale and in documentation and status labels, until claims were bound to mechanisms by single sources of truth and tests. This finding qualifies any suggestion that a rebuild, or a single audit, settles the matter.

Fifth, it states **seven propositions** connecting these observations to theories of signalling and credence goods [14], [15], [16], organisational decoupling [17], [18], symbolic management in new ventures [19], [20], and the economics of information security [21], [22].

The paper is the second in a series. The first studied a sister venture built under hard resource constraints and introduced the idea of *verification debt*: the accumulating gap between what a system was built to do and what anyone has checked that it does [23]. The present case shows what verification debt looks like when it is total, and what it takes to pay it down.

The remainder of the paper is organised as follows. Section 2 reviews the relevant literatures and identifies the gap this paper addresses. Section 3 describes the method, data and case. Section 4 reports findings on the first version, the audit, the rebuild and its aftermath. Section 5 develops the frameworks and propositions. Section 6 discusses implications, limitations and directions for research, and concludes.

2. Literature review

Simulated completeness sits where five literatures meet:

- the economics of quality uncertainty;
- institutional theory on the decoupling of appearance from activity;
- entrepreneurship research on symbolic management;
- research on AI-assisted software development;
- research on software verification and on human oversight of automation.

The case domain, bot verification, has its own literature, which is reviewed last.

2.1 Quality uncertainty, credence goods and security markets

Akerlof's analysis of the market for "lemons" showed that when buyers cannot observe quality, sellers of low-quality goods can pass them off as high-quality. Prices then fall to reflect average quality, and good sellers may leave the market [14]. Spence showed how costly signals can separate types when the cost of the signal is correlated with the quality it signals [24]. The crucial condition is the correlation. A signal that is equally cheap for every type separates nothing.

Darby and Karni distinguished goods by when their quality becomes observable. Search goods can be judged before purchase and experience goods after use. *Credence* goods cannot be judged even after use without expert help, and these invite fraud [15]. Dulleck and Kerschbamer's review of credence-goods markets sets out the forms that fraud takes: overtreatment, undertreatment and overcharging. It also shows that liability, verifiability and reputation determine whether markets discipline it [16].

Security products are a canonical credence good. A buyer who installs a bot defence and sees no attacks cannot easily tell whether the defence worked or whether nobody attacked. Anderson argued that information security fails as often for economic reasons as technical ones: misaligned incentives, and markets in which buyers cannot tell good security from bad, so that vendors compete on features and assurances rather than on protection [21]. Anderson and Moore extended the argument into a research programme on the economics of security [22]. Schneier's phrase "security theater" describes measures that provide the feeling of security without the reality [25]. Each of these accounts takes as given that a product either contains a mechanism or does not. None asks what happens when the cost of producing the *appearance* of a mechanism falls towards zero. In signalling terms, that is precisely the condition under which a signal stops separating types.

2.2 Decoupling: formal structure as ceremony

Meyer and Rowan argued that organisations adopt formal structures (job titles, procedures, certifications) because those structures confer legitimacy, not because they make work more efficient. Where the formal structure conflicts with efficient activity, organisations *decouple* the two. They maintain the structure ceremoni-