COLDCODE SI LABS LIBRARY

The Mechanism Auditor: How to Review a Product That Runs Without You

Courage Garikai
Coldcode Si Labs Library · Harare
November 2026

Part One — The Witness Is Gone

1. The demo was a promise

For as long as software has been sold, it has been sold by demonstration. Someone sits with someone else and makes the thing work. The buyer's question — *does it do what you say?* — is answered by watching it happen, and the watching is the proof, because the person making it happen is the person who built it, and in thirty years of practice we have learned roughly how much they might be tempted to fudge.

That arrangement had a hidden requirement, and nobody ever wrote it down: **a human had to be present at every act**. The demonstration was evidence because a person was in the loop, doing the thing in front of a witness. If the demonstration was honest, the software was at least capable of the act. The reviewer's job was to spot the trick.

THE MECHANISM AUDITOR

The Mechanism Series · first edition

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 transmitte
form or by any means without the prior written permission of the publisher, except in the ca
quotations in a review or certain other noncommercial uses permitted by copyright law. The
frameworks, worksheets and checklists in this publication are meant to be used, copied and
by the reader.

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

COLDCODE SI LABS LIBRARY · Harare, Zimbabwe

The trick is still possible, but it is no longer the interesting problem. The interesting problem is that the acts which matter now happen when nobody is watching. Software commits changes to itself on a schedule, publishes to accounts, sends invoices, rotates credentials, and reports its own success. By the time a human looks, the act is over, and all that remains is what the software chose to tell you about it.

I run seven small software businesses from one desk in Harare, with agents doing most of the building and one person — me — accountable for all of it. There is no QA department, no second pair of eyes, and no meeting in which someone says "show me again". Every failure I describe in this book happened in that estate, in one week of ordinary operation, and every one of them was invisible from the demo. They were visible only in the mechanism, and the mechanism is what this book teaches you to read.

2. What a green tick actually says

A build turns green. Everyone relaxes. This is the single most expensive habit in modern software, and it survives because of a small linguistic trick: we say *the build passed*, as though the build were the product.

A green tick says one thing: **a specific job, running a specific set of checks on a specific revision, exited zero**. It does not say the product works. It does not say the checks could have failed. It does not say the revision you deployed is the revision that ran. It does not say anything happened in the world outside the runner.

Four separate claims are being conflated, and each of them can be false while the tick stays green:

- **Coverage.** The checks might not touch the thing that broke.
- **Capability.** The checks might be incapable of failing, because of an exception, a skip, a dry run, or a missing prerequisite.
- **Currency.** The revision that ran might not be the revision you shipped.

- **Effect.** The job might have finished before the effect it was meant to have occurred, or the effect might have occurred somewhere you are not looking.

Every failure in Part Three is one of these four, and none of them is a bug in the usual sense. They are all failures of the *story the tick tells*.

So the audit begins with a decision that feels petty and is not: **stop treating a green tick as evidence, and start asking what evidence would look like.** The rest of this book is that question, answered five ways.

Part Two — The Five Artefacts

An audit that produces a score has produced a second thing that cannot fail. An audit that produces artefacts has produced something a stranger can check. Five of them, in order, each worthless without the ones before it.

2a. The economics of silent failure

Silent failures are cheap in the way that a leaking roof is cheap: the cost is not the leak, it is the interval before anyone looks.

Every failure in this book has the same shape. A mechanism did not do what it claimed, a signal said it had, and the interval between those two facts was measured in days. In that interval the product was **confidently wrong**, and confidence is what makes a wrong thing expensive: a customer bought, an editor published, a payment ran, a promise went out attached to a name.

Three numbers decide whether an audit is worth its price, and none of them is the auditor's hourly rate:

- **The cost of the failure met by a customer** — a refund, a replacement, an apology, and the specific damage of having been confidently wrong to someone who trusted you.
- **The cost of the interval** — how many of those failures fit into the time between your looks.
- **The cost of looking** — an hour, a day, a job that runs every thirty hours.

For a product that acts a few times a day, the interval is the whole story. A control that reports weekly and a control that reports hourly differ by a factor of a hundred and sixty-eight in exposure, and no amount of dashboard polish closes that gap.