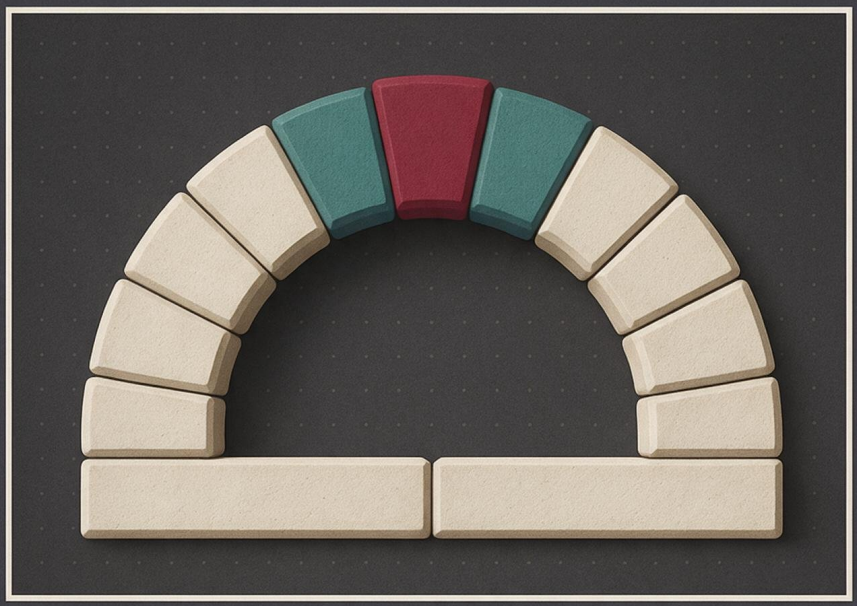
COURAGE GARIKAI



SLOW SOFTWARE

Building Technology That Lasts

COLDCODE SI LABS · HARARE

© 2026 Courage Garikai. All rights reserved.

Published by Coldcode Si Labs, Harare, Zimbabwe.

First edition, November 2026.

The moral right of the author has been asserted. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form without the prior written permission of the publisher, except for brief quotations in reviews.

Contact: couragegarikai@atomicmail.io · library.coldcode-si-labs.workers.dev

For the engineers who maintain the bridges.

For the developers who fix bugs at 3 AM without complaining about it on Twitter. For the sysadmins who keep services running for decades. For the documenters who write the READMEs that no one reads but everyone depends on.

And for the bridge-builders of Norton, 1937.

Slow Software

Building Technology That Lasts

Courage Garikai

ColdCode Si Labs · Harare

About this book

The technology industry moves fast. Frameworks are released weekly. Startups pivot monthly. Apps ship updates daily. Social media moves in minutes. Messages arrive in milliseconds. The entire industry is built on a single assumption: faster is better.

This book argues the opposite. Not that fast is always bad, or that speed has no value, but that we have let the pursuit of speed — fast deployment, fast growth, fast iteration, fast features, fast

responses, fast scaling — override every other value in software: reliability, longevity, maintainability, accessibility, thoughtfulness, durability, craftsmanship, quiet, stability, and the ability to build software that outlives its builders.

I call the alternative slow software. Slow software is not software that loads slowly or runs slowly for users. It is software that is built slowly, deliberately, with attention to longevity rather than velocity. Slow software is fast for the user but patient in its construction. It is software designed to be used for decades rather than quarters. It is software that prefers the boring and tested over the novel and trendy. It is software that respects the user's time and attention rather than racing to capture it. It is software built by teams and individuals who prioritize understanding over speed, stability over features, and longevity over growth.

We have forgotten how to build slow software. The current incentives of venture capital, startup culture, continuous deployment, agile sprints, growth hacking, and "move fast and break things" have trained two generations of software

developers to ship fast, iterate quickly, deprecate ruthlessly, and treat every piece of technology as disposable. The result is software that breaks constantly, requires constant updates, stops working when the company is acquired or shuts down, demands endless user attention, and leaves nothing durable behind.

This is not a Luddite manifesto. I am not arguing against progress, against new technology, or against speed where speed serves the user. I am arguing for balance — for a discipline of software construction that values slowness where slowness produces better outcomes:

- Slow in design, to understand problems before coding solutions.
- Slow in construction, to write code that can be maintained for years.
- Slow in adoption, to choose technologies that will be supported in a decade.
- Slow in release, to test thoroughly before users depend on the software.

Sample provided free for review. The complete book is available at https://library.coldcode-si-labs.workers.dev/