COLDCODE SI LABS LIBRARY

The Tight Bucket

Designing constraint-native digital businesses

Theories, frameworks and one working case for founders, directors and strategists

Courage Garikai

Funded by Coldcode Si Labs



FIRST EDITION · 2026

THE TIGHT BUCKET

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

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

COLDCODE SI LABS LIBRARY · Harare, Zimbabwe

Contents

About this edition 4

PART I · The new economics

1 · The bucket and the flood 10
2 · When building is cheap, what is scarce? 15
3 · Business models as activity systems 20
4 · Constraints as design material 26

PART II · The value engine

5 · Proof before pitch 33
6 · The zero price and the free tier 38
7 · One engine, four audiences 43
8 · Pricing architecture 48
9 · Borrowed commerce 53

PART III · Distribution without an audience

10 · Be the flower 59
11 · Inbox physics 64
12 · Armed but dormant 69
13 · Outbound with proof, replies by hand 73
14 · Loops and partners 78

PART IV · The operating model

15 · The autonomy boundary 84
16 · Quota-bounded autonomy 89
17 · Unit economics at the meter 95
18 · Verification debt 100
19 · Tenancy at purchase 105
20 · The cockpit 110

PART V · Learning and strategy

21 · Experiments without vanity 116
22 · Strategy as choice under constraint 121
23 · Risk and dependence in borrowed infrastructure 126

24 · A ninety-day programme 131

TOOLKIT · Frameworks, guide, glossary and references

25 · Framework compendium 138
26 · Facilitator's guide 146
27 · Glossary 151
28 · References 156

About this edition

First edition · 2026

This book is written for digital entrepreneurs, SaaS founders, company directors and anyone responsible for business model design, strategy or the direction of a digital business. It is a discussion book. Each chapter sets out a question, the theory that helps answer it, a working case, a framework you can use, the points on which reasonable people disagree, questions for discussion and a short exercise for your own business.

The case is TightBucket, a website "revenue-leak" auditing service built in three days in October 2026 by its owner working with AI coding agents. TightBucket was commissioned by the same owner who commissions the work of Coldcode Labs Think Tank. It is therefore a bounded case, not an independent evaluation. At the time of writing TightBucket had **not launched commercially**: it had taken no payments and has no customer results to report. Nothing in this book should be read as evidence that its business model succeeds commercially. What the case offers is unusual visibility into how a business was *designed* and *governed* under hard constraints, recorded in its code, documents, defect log and usage data.

Throughout, three kinds of statement are kept apart:

- **Research findings** are cited to their sources.
- **Measured facts about the case** come from its repository, its usage data and its providers' published terms as of October 2026.
- **Design targets, constructed examples and proposed frameworks** are labelled as such. They are recommendations and propositions, not proven laws.

Prices, quotas and platform terms change. Figures quoted from providers are as published in October 2026.

No professional biography, endorsement, ISBN or independent audit is claimed. The author and funding credits are those specified by the commissioning owner. The case venture was built with AI coding agents, and AI tools assisted in drafting

© 2026 Courage Garikai. All rights reserved. Published by Coldcode Si La
Library, Harare. No part may be reproduced without written permission,
the frameworks and exercises, which are meant to be copied.

The promise of this book

For most of the history of software businesses, the hard part was building. You needed engineers, months and money before anyone could judge whether the idea deserved them. Strategy was largely about deciding what to build with scarce engineering capacity.

That constraint is weakening fast. A single owner working with AI agents can now produce a working, tested, multi-audience product in days. Cloud providers' free tiers are generous enough to run it. Merchant platforms will handle checkout, tax and payouts for a percentage of each sale. The cash cost of *starting* is approaching zero.

This does not make business design easier. It moves the difficulty. When building is cheap, the scarce things are attention (yours and your prospects'), trust, reputation in borrowed channels, and the ability to check that what was built behaves as intended. When infrastructure is free, its rules (quotas, sending limits, platform terms) become the real boundaries of the business. A founder who ignores them finds out about them in production.

The central argument of this book is that such constraints should be the *starting material* of a business model, not obstacles discovered later. We call a business designed this way *constraint-native*. The image in the title is the bucket that holds what it catches: a business that leaks nothing it has earned (no wasted attention, no burned reputation, no surprise bills) because its limits were designed in from the start.

Who this book is for

- **Founders and solo operators** who can now build almost anything and need to decide what to build, how to sell it and how to run it without a team.
- **SaaS founders and product leaders** designing free tiers, pricing ladders and acquisition systems.

- **Directors and board members** overseeing products that are increasingly built and operated by AI, who need ways to ask the right governance questions.
- **Strategists, consultants and educators** who want a contemporary case and a set of frameworks to teach with.

No programming knowledge is assumed. Technical details appear only where they change a business decision, and are explained when they do.

How to use this book

Each chapter follows the same pattern:

1. **An opening question** that the chapter tries to answer.
2. **Theory**: the research and ideas that bear on the question, cited so that you can read further.
3. **The TightBucket case**: what the case did, and what it shows and does not show.
4. **A framework or tool** you can apply.
5. **Points of debate**: where the argument is contestable.
6. **Discussion questions** for a reading group, a class, a leadership team or your own reflection.
7. **Apply it**: a short exercise for your own business.

The chapters can be read in order or by part. Groups meeting weekly can take one part at a time over five sessions, or one chapter at a time over a semester. Chapter 25 collects every framework in one place. Chapter 26 is a facilitator's guide, Chapter 27 is a glossary and Chapter 28 lists the references.

The structure

- **Part I · The new economics** (Chapters 1–4) explains what changes when building is cheap, and introduces constraints as design material.
- **Part II · The value engine** (Chapters 5–9) covers proof-first offers, free and freemium, serving several audiences with one engine, pricing architecture and selling through borrowed commerce.

- **Part III · Distribution without an audience** (Chapters 10–14) covers inbound pull, the physics of the inbox, dormant capacity, honest outbound and growth loops.
- **Part IV · The operating model** (Chapters 15–20) covers where autonomy should stop, governing an autonomous business against quotas, unit economics at the level of the meter, verification debt, customer tenancy and the owner's cockpit.
- **Part V · Learning and strategy** (Chapters 21–24) covers experiments, strategy as choice under constraint, the risks of borrowed infrastructure and a ninety-day programme.

The case in one page

TightBucket audits a business's public website for "revenue leaks": slow pages, unclear pricing, broken forms, missing email authentication. Given an address, it scores the site out of 100 and ranks the fixes. It serves four audiences under one brand (e-commerce merchants, SaaS founders, sales directors and agencies), each with its own audit page and free tools, ten free tools in all.

Its paid ladder runs from free use with no sign-up, through a free account and paid plans at $39 and $149 a month, to done-for-you remediation from $300. A store plan at $19 a month is sold inside an e-commerce platform's app store.

The owner set the constraints before anything was built:

- zero budget;
- checkout entirely on a merchant platform;
- email through a free mailbox;
- four audiences under one roof;
- autonomous operation with a human only for important work;
- no automation of a professional network whose terms forbid it;
- no existing audience;
- customers who seek the business out;
- and no more than 20% of any free cloud quota or free automation minutes, "so that I never need to worry about quotas".

The result, built between 26 and 28 October 2026, is a single serverless application on a free plan with about 7,800 lines of application code, about 6,450 lines of distribution code and 396 automated tests. It has:

- a quota governor that measures every database row and AI unit and cross-checks the provider's own figures;
- two email lanes;
- an owner's cockpit;
- automatic team workspaces for paying customers;
- a distribution engine of more than thirty publishing channels, built and waiting for accounts.

Two line-by-line reviews found and fixed 43 defects. At its observed peak it used less than 2% of any free quota. At the end of the observation window it had 19 audits on record, 9 outreach messages prepared for the owner, no sales and one member: the owner.

That is the whole case. The rest of the book asks what it teaches.

PART I

The new economics

The Tight Bucket

1 · The bucket and the flood

> **Opening question.** If almost anyone can now build a working software product in days, what still separates a business from a pile of software?

In October 2026 one owner, working with AI coding agents, built a complete multi-audience software business in three days. It had a scoring engine, ten free tools, a paid ladder, two email systems, a customer portal, team workspaces, an owner's console and a distribution engine for more than thirty channels. It had 396 automated tests. It had taken no money and spent none.

Ten years earlier that list would have described a funded start-up's first year. The question this chapter asks is what such a capability does to business design. Our short answer is that it creates a *flood*, and that the businesses which last will be the ones that behave like a good *bucket*.

The cost of building has collapsed

The evidence that AI changes the cost of building is now substantial, though uneven. In a controlled experiment, programmers given an AI pair-programmer completed a bounded coding task 55.8% faster than those without (Peng et al., 2023). In professional writing tasks, access to a chat assistant cut the time taken and raised the assessed quality of the work (Noy & Zhang, 2023). In a large customer-support operation, an AI assistant raised the number of issues resolved per hour by 14% on average, and by 34% for novice and less-skilled workers (Brynjolfsson, Li & Raymond, 2025).

The gains are not universal. A field experiment with management consultants found large improvements on tasks inside the AI's capability "frontier", but *worse* performance on a task just outside it. The authors call this a "jagged technological frontier" (Dell'Acqua et al., 2026). The tool is excellent at some things and confidently wrong at others, and the boundary is not visible from the outside.

Infrastructure has followed the same path. Serverless computing lets a developer run code without provisioning or managing servers, paying only for what is used (Jonas et al., 2019). Many providers now give away enough capacity to run a small business at no charge. The provider used in our case, for example, allows 100,000 requests and 100,000 database row writes a day on its free plan.

Put the two together and the cash cost of *starting* a software business approaches zero.

A wealth of products creates a poverty of attention

Herbert Simon made the essential point decades before any of this: "a wealth of information creates a poverty of attention" (Simon, 1971). What information consumes, he observed, is the attention of its recipients.

When building is cheap, products, features, landing pages, cold emails and social posts all multiply. The supply of *things competing for attention* rises much faster than the attention available to absorb them. Every inbox, feed and marketplace fills up. That is the flood.

Information goods have long had high fixed costs and near-zero marginal costs, which pushed their economics towards versioning, bundling and lock-in (Shapiro & Varian, 1999). What is new is that the *fixed* cost of creating the good is falling too. When both fixed and marginal costs fall, the price of the product is no longer the main barrier between a buyer and a seller. The barrier is the buyer's attention and trust.

Why a bucket?

A bucket is a modest image for a business, chosen deliberately. A bucket does not make water. It holds what it catches, and its quality is judged by what it does *not* lose. A leaky bucket can be filled endlessly and still end up empty.

Businesses in a flood leak in four characteristic ways:

- **Attention leaks:** a prospect looks once and never returns, because the first contact asked for something before it gave anything.