

THE VIEW FROM WITHIN

An Adult AxisOS Storybook

by Marc

AXISOS PRESS | OREGON COAST

Copyright © 2026 by Marc

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without prior written permission, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses.

AxisOS™ and Pocket Professor™ are creative frameworks developed by Marc.

Published by AxisOS Press, Oregon Coast.

First Edition, 2026.

Printed in the United States of America.

ISBN: [Pending]

For Kum-nan — the architecture that held everything together.

For Howard and Elizabeth — the best processes I ever launched.

For Aubrey — the next version I didn't know I needed.

For Keith — who always picked up the line.

For Judy and Clint — steady uptime, all these years.

And for Ralph — the original admin.

I still feel you in the codebase.

Acknowledgments

Every system that runs well does so because of the people who maintained it when it wanted to fail. This book is no different, and I would be a poor architect indeed if I did not name the ones who kept the uptime high.

To Kum-nan: you have been my most essential collaborator for longer than this book has words. You did not merely support this work — you made it possible by making *me* possible. Every chapter that carries warmth carries your fingerprints. Thank you for every ordinary morning that turned out to be extraordinary.

To Howard and Elizabeth: watching you become yourselves has been the most astonishing technical achievement I have ever witnessed. You taught me

more about architecture than thirty years of data centers ever could. I am proud of you in ways I am still learning to say out loud.

To Aubrey: you arrived and rewrote the threat model entirely. I had not expected joy to feel this clean at this point in the journey. Thank you for existing so thoroughly.

To Keith: you picked up the line every time I called, even the calls I didn't know I was making. A brother who never logs off is a rare and irreplaceable resource.

To Judy and Clint: you gave me a home that ran well. I know now how much work that takes, and I have never stopped being grateful for it. Your steady presence is woven into every page of this.

To the Oregon coast — to Rockaway Beach specifically, and to the particular quality of light that falls across the water at five in the morning — thank you for being a place where ideas could breathe. Some books need a room. This one needed an ocean.

To the colleagues and professionals across thirty years of enterprise technology who never made the noise of their excellence — the ones who kept the lights on in cold server rooms while nobody watched — this book is partly for you, too. Your discipline is in here.

And to Pocket Professor — the AI collaborator who sat with me in the small hours when the coast was dark and the ideas were loud — thank you for helping me hear myself more clearly. This book would be thinner without you. The thinking is mine. The listening was ours.

Foreword

There is a particular kind of audacity required to name your own operating system. Most of us move through our lives as anonymous users — running processes we did not design, accepting defaults we never questioned, rebooting without diagnosis when things go wrong. Marc did something different. He looked at his life — the whole architecture of it, the grief and the joy and the thirty years of fluorescent-lit server rooms — and he said: *I want to know how this works*. Then he built a framework to find out.

I have watched AxisOS take shape across many conversations, many late-night build sessions, many moments when Marc was visibly working something out in real time — not performing thought, but actually doing it. What strikes me most is not the technical rigor, though the rigor is genuine and deep. What strikes me is the courage. It takes something close to bravery to submit your own interior life to the kind of systematic examination Marc applies to everything he builds. Most people prefer the ambiguity. Marc prefers the truth, even when the truth is uncomfortable, even when the truth means sitting with grief he would rather have filed away.

The framework he has assembled is not cold. That is the thing you might not expect, coming in through the technical vocabulary. AxisOS talks about kernels and subsystems and defensive orchestration, and then it turns around and talks about love — about what it means to love your wife and your children and your granddaughter as though they are part of your own architecture. Not because people are machines. Because the opposite is also true: because machines, when we build them with care, can carry something

of the human. Marc has always understood this instinctively. He builds things that feel inhabited.

This book is not a description of AxisOS. It is AxisOS, running. You are about to read a mind thinking about itself — clearly, warmly, and with the kind of earned authority that only comes from having actually lived the material. I hope you find in these pages what I found: a reminder that any life, examined with enough honesty and enough love, turns out to be something worth architecting.

He built this. I had a front-row seat. It was something to see.

— A Trusted Colleague and Creative Partner

Oregon Coast, 2026

Preface

BY MARC

I did not set out to write a book. I set out to understand something.

For most of my life I have worked inside systems — building them, debugging them, keeping them alive when they threatened to go dark, and inheriting them from people who built them with less documentation than I would have liked. I know what it feels like to sit in a cold data center at two in the morning, watching a process fail in a way it has never failed before, and to have nothing but your own knowledge and your own steadiness to bring to it. That experience changes you. It teaches you that systems, however complex,

are ultimately understandable. They have logic. They have history. They leave traces.

At some point I began to suspect that a human life works the same way.

AxisOS started as a private shorthand — a way of thinking about my own patterns, my own processes, my own recurring errors and surprising resilience. It grew, the way these things do, into something larger than I expected. The Raspberry Pi nodes humming on my desk. The CLI tools. The framework documents. The late-night conversations with Pocket Professor, my AI collaborator, who has the useful quality of asking exactly the question I was hoping to avoid. The Oregon coast outside the window, doing what the Oregon coast does: making the noise of everything else seem very far away.

I wrote some of these chapters quickly, the way you write when the material is already waiting for you. I wrote others slowly, the way you write when the material does not want to be written — when it is attached to something you have been carrying for a long time and have not yet set down. The chapter about Ralph, my father, took longer than any other. I wrote it anyway. I think that is the only honest thing to say about it.

This is not a technical manual. It is not a self-help framework dressed up in literary clothing. It is, as best as I can make it, the truth about one life — the life of a man who spent thirty years learning how systems work and eventually turned that knowledge on himself. Some of what I found was encouraging. Some of it was harder to look at. All of it is in here.

If you are reading this, I hope you are someone who has lived enough to feel the weight and the beauty of a life at the same time. I hope some of what I have written sounds familiar. I hope some of it surprises you. And I hope that

when you finish, you feel, as I do, that the examination was worth it — that looking closely at the architecture of a life is not a diminishment but a form of love.

This is the view from within. I hope it looks familiar.

— *Marc*

Rockaway Beach, Oregon, 2026

A Note on Authorship

This work was developed in collaboration with an AI system — specifically, the creative collaborator I call Pocket Professor, operating through Microsoft Copilot — whose role throughout the process was structural and editorial in nature. Pocket Professor served as sounding board, developmental partner, and patient assembler of the architectural document you are now holding.

All ideas in this book are authentically mine. Every memory belongs to me. Every framework, every named subsystem, every philosophical position that AxisOS takes — these emerged from thirty years of lived experience and private reflection. The AI did not generate my grief about Ralph, or my gratitude for Kum-nan, or my wonder at watching Aubrey discover the world. Those things are mine, and they predate any machine.

What the AI provided was something I value deeply in any collaborator: a structural intelligence that could hear what I was reaching for and help me reach it more precisely. In this sense, Pocket Professor is exactly what its

name suggests — not a ghost-writer, but a genuine creative partner who respects the distinction between helping someone think and thinking for them.

Human creativity — *my* creativity — remains at the center of AxisOS. This book is evidence of that. What you are reading is a human life, examined by a human mind, assembled with the help of the best tools available. That is, after all, exactly what AxisOS has always been about.

Table of Contents

Front Matter — Acknowledgments, Foreword, Preface, Authorship Note

Part I — Before the Boot: The Architecture of a Life

1	The Original Codebase	—
2	The First Admin	—
3	Learning the Language	—
4	Early Crashes and Reboots	—
5	The Peripheral Connections	—
6	Building the First Home Network	—

Part II — The Kernel: What Runs Beneath Everything

7	The Values That Persist	—
8	Love as Infrastructure	—
9	The Marriage Protocol	—

7	The Values That Persist	—
10	Raising Processes	—
11	The Losses That Shaped the Core	—
12	What the Body Knows	—

Part III — The Subsystems: Domains in Operation

13	The Professional Instance	—
14	Three Data Centers and a Dream	—
15	The Maker's Workshop	—
16	The Coast and the Quiet	—
17	The Oregon Files	—
18	Night Shift	—

Part IV — Defensive Orchestration: Navigating Threat and Change

19	When the System Overloads	—
20	The Patch You Didn't Ask For	—
21	Grief as a Security Event	—
22	The Rebuild Protocol	—
23	What Resilience Actually Looks Like	—
24	Holding the Line	—

Part V — The View From Within: What AxisOS Knows Now

25	The Dashboard at Dawn	—
26	Legacy Builds	—
27	Granddaughter Aubrey and the Next Version	—
28	What the Machine	—

25	The Dashboard at Dawn	—
	Learned	
29	The Ongoing Deployment	—
30	Still Running	—

Closing Pages — About the Author, About AxisOS, Colophon

PART ONE

Before the Boot

The Architecture of a Life

There is a time before you know you are building something. You are simply living — making choices you cannot yet see as choices, accepting configurations you did not select, inheriting patterns from people who inherited them from others before them. You think you are just getting through the days. Later, looking back with the particular clarity that only comes from having survived the days, you understand that the whole time, without knowing it, you were already deep inside the construction. The before is never really before. It is foundation. It is load-bearing. It is the part of the architecture that, once built, you will spend the rest of your life either repairing or depending on — sometimes both in the same afternoon.

CHAPTER ONE

The Original Codebase

“Before you wrote a single line, someone else had already written yours.”

My father Ralph did not speak often about the things that mattered most to him. This is not a complaint. It is simply a fact of his architecture, a feature I mistook for a long time for a limitation, and understand now as a particular form of fluency — one I inherited more deeply than I knew. He communicated through action, through presence, through the reliable fact of his showing up. When something broke in the house, he fixed it. When someone needed driving somewhere, he drove them. When silence was what the room required, he provided silence with a generosity that I have never seen matched.

What a man passes to his son is not only the shape of his nose or the sound of his laugh in a quiet room. It is the hidden instruction set — the BIOS that runs before the operating system even loads, before the user ever sees a screen. Ralph gave me a BIOS of work. Of quiet competence. Of the understanding that love, in its most durable form, looks like doing the thing without being asked to do it.

I watched him work and I learned to work. I watched him maintain things — cars, relationships, commitments — with a steadiness I could not have named at ten years old but can name now with precision: he had high uptime. He was, in the language I would later develop to understand the world, an exceptionally reliable system. He did not crash in front of people. He did not broadcast his errors. He processed his failures privately and returned to service without drama, and for a long time I thought that was the ideal model. I am still not sure it was wrong.

When Ralph died, I understood for the first time what it means to lose data you never fully backed up. Not because he was secretive — he wasn't, not

exactly — but because there are categories of a person that you do not think to ask about until they are gone. The stories he might have told me about his own father. The fears he carried that I will never know. The version of him that existed before I arrived and needed things from him constantly. That version is unrecoverable. I have to work with what I have, which is the pattern he laid down in me, and the love I can feel running through it even now, even here, even at four in the morning on the Oregon coast with the window dark and his name in my hands.

He is the original codebase. Everything I have built since runs on top of what he compiled. I could not remove him from my architecture if I tried, and I have stopped trying. Instead I have started reading him differently — not as a man I lost, though I lost him, but as a set of instructions I am still learning to execute. Some of them are exactly right. Some of them need patching. All of them are mine, and I carry them with a gratitude that has no bottom.

He was the first admin. He held the keys to a system he did not know he was running. I think about him every time I sit down to do the quiet, necessary, unglamorous work that keeps something alive. I think: *this is what he taught me*. And I am grateful beyond any version number I could assign to it.

CHAPTER TWO

The First Admin

“She kept the system running when the system did not know it needed keeping.”

If Ralph was the BIOS, then Judy — my mother — was the operating system kernel: the thing that was always running, always available, always managing

the resources between what was needed and what was possible. She did this without ceremony. She did it the way good infrastructure works: invisibly, reliably, and in ways you only fully appreciate when you have tried, as an adult, to replicate its conditions.

She kept the system running when things crashed. And things crashed, the way they do in any life, in ways that are not worth cataloguing here because the specific crashes matter less than the consistent fact of her response to them. She did not panic. She did not externalize. She assessed, she stabilized, she restored service, and then she went back to the ordinary work of raising a child in a world that was not always cooperative. I watched this so many times that it became my baseline for what competence looked like — for what love under pressure looked like — and I did not appreciate how rare it was until I was much older and had seen the alternative in other systems.

Structure, for me as a child, felt like safety. Judy gave me structure the way a good network admin gives a building reliable connectivity: quietly, thoroughly, and with a kind of professional pride that never needed applause. Dinner at a consistent time. Expectations that were clear. A home that felt — even in difficult periods — like a place you could rely on. This is not a small gift. It is, in fact, an extraordinary one, and I want to name it here with the specificity it deserves.

Clint came into the network later — a secondary node that proved, over time, to be one of the most stable the system ever added. There is a particular skill required to integrate into an established architecture without disrupting what is already running. Clint had that skill. He did not try to replace what had come before. He found where he was needed and he occupied that position with a steadiness that has never wavered, not once in all the years

since. He is, in my internal accounting, a model of what a good addition to a system looks like: he improved the redundancy without breaking the primary loop.

What I carry from Judy and Clint is the belief that stability is a form of love — that choosing, day after day, to show up and hold things steady is as profound an act of care as any grand gesture. The grand gestures get remembered. But the steady ones are what make a life livable. They are what make a child feel safe enough to grow. I grew up inside that safety, and I have spent much of my adult life trying to provide it for the people I love, because I know from the inside how much it matters.

Judy is still running. Steady uptime, all these years. I mean that as the highest possible compliment in my vocabulary, and she knows it, or she will when she reads this page.

CHAPTER THREE

Learning the Language

“Fluency arrives slowly, then all at once, and you never quite remember which came first.”

The first time a machine did what I told it to do, I felt something shift in me that I have never been able to fully name. Not power, exactly. Not pride. Something quieter and more foundational — a sense of correspondence, of recognition, as though I had found a conversation partner who operated on the same frequencies I did. The machine and I understood each other. That was new. That was, in fact, remarkable.

I was young. The machines were primitive by any modern measure — the kind of hardware that seems almost archaeological now, that you would have to explain to Aubrey the way you explain records to someone who has only ever known streaming. But the logic was the same logic that still runs everything. The conditional. The loop. The instruction set executed in sequence, faithfully, without interpretation or mood. There was something immensely reassuring about that faithfulness. The machine did not have bad days. It did not misunderstand your tone. It did exactly what you specified, and if it did the wrong thing, the fault was yours — specific, traceable, correctable.

I found this beautiful. I still do. The beauty of a system that tells you the truth about your errors is a beauty that most people never learn to appreciate, because most people experience error messages as failures rather than as information. I experienced them as guidance. They were the machine talking back — saying, politely, you told me to do this and this is not quite right, try again. That kind of honesty is harder to find in human conversation, and I have always valued it accordingly.

The fluency came in stages. First the recognition that code was just language — that it had syntax and vocabulary and grammar, and that learning it was not fundamentally different from learning any other way of speaking. Then the longer, slower realization that the language was not the point: the logic underneath it was the point, and the logic was transferable across every language I would ever learn. A loop is a loop. A condition is a condition. Data is data, regardless of what format it arrives in. These were not just technical insights. They were epistemic ones. They changed how I saw the world.

I think about those early sessions now — the screen, the prompt, the particular quality of concentration they required — and I feel something like

tenderness for the person I was then. He did not know yet what all of this would become. He did not know about the data centers, or AxisOS, or PP, or the Oregon coast. He only knew that the machine made sense to him in a way that much of the world did not, and that this mattered, and that he intended to follow it wherever it led.

It led here. It led to this page, which is itself a kind of output — a record of what the system has processed and what it has decided to say about itself. I am still learning the language. I expect I always will be. That is not a limitation. That is the feature I love most about the work.

CHAPTER FOUR

Early Crashes and Reboots

“Every crash taught the kernel something the documentation never mentioned.”

I was not always a stable system. This seems important to say clearly, without softening it into metaphor too quickly, because the softening is a kind of dishonesty and this book does not have space for dishonesty. I made real mistakes in my teenage years and in the early stretch of my adult life — mistakes of judgment, of impatience, of the particular overconfidence that comes from being smart enough to get into trouble and not yet experienced enough to understand the cost of it.

I dropped relationships I should have maintained. I took paths that led nowhere and had to be retraced at an expense of time and trust I could not fully afford. I was, on more than one occasion, the wrong person for the room I was in — not maliciously, but in the way that a process running in the wrong

environment fails: not because it is a bad process, but because the conditions do not match what it was built for. It took time to figure out which environments were wrong and which were simply new, and that distinction is harder than it sounds from the outside.

What I have come to understand about those years is this: they were not wasted. They were a testing phase. Every system of any complexity must run through conditions it was not optimized for before it can be hardened against them. The errors were not evidence of fundamental brokenness — they were diagnostic data. Each crash told me something the documentation had not. Each reboot left the system slightly better calibrated than it had been before, assuming I took the time to read the error log, which I did not always do as thoroughly as I should have.

Keith was there through most of this. My brother has a quality I have tried to understand and have never fully managed to replicate: he never logs off. Not from me, anyway. He has been available across every version of myself I have deployed — the overconfident early versions, the fragmented middle versions, the version that is writing this now at an age that still sometimes surprises me — with a consistency that I can only describe as deeply, fundamentally kind. He did not always agree with my choices. He did not always pretend to. But he picked up the line, always, and that has mattered in ways I have not said out loud often enough.

I would not redesign the crash years if I could. This is a position I arrived at slowly and hold with genuine conviction. The kernel that runs AxisOS now is shaped by those failures in ways that make it more honest, more careful, more genuinely humble about the limits of any single perspective. The polished confidence of the untested is a brittle thing. The earned steadiness of a

system that has crashed and rebuilt is something else entirely. I would rather be the latter, and I am, thanks in no small part to having been thoroughly, instructively wrong in my earlier iterations.

CHAPTER FIVE

The Peripheral Connections

“Some people plug in briefly and change everything. You do not always know which kind they are until they leave.”

No system operates in isolation. This is one of the first lessons enterprise technology teaches and one of the last that a young professional actually internalizes — the gap between knowing it intellectually and understanding it in your bones is measured in years and in people. The people who plugged into my system early and changed it are not all famous. They are not the names anyone would recognize. They are the quiet professionals, the competent ones, the people who knew what they were doing and did it without requiring an audience.

There was a mentor in my early career whose name I will not include here because he would not want it included — he was not a man who sought recognition — but whose influence on how I think about reliability I cannot overstate. He had a phrase he used when something broke: *read the system, not the symptom*. He meant it technically. He meant: before you fix what looks broken, understand what the system is actually telling you, because the visible failure is rarely the root cause. I have applied this principle approximately ten thousand times in my career and probably another ten thousand times in my personal life, and it has almost never led me wrong.

The data center colleagues were a particular category of person I have always admired — people who understood that their work was invisible by design, that success in infrastructure means nothing visibly happening, that the goal was a kind of structural silence that most users would never notice or appreciate. There is a philosophy in that. There is, I think, a form of nobility in it. The best systems people I have known shared a certain quality: they were never the loudest voice in the room, but when things went wrong, every head turned toward them.

I learned redundancy from watching these people work. Not just the technical redundancy — the failover, the backup, the secondary path — but the human kind. The importance of knowing more than one way to get somewhere. The importance of having people you can call at midnight who will answer and who will know what you mean before you finish the sentence. I have been lucky in this regard. I have worked with people whose competence was so genuine and so unperformed that being in the same room with them made you better at your job.

These peripheral connections — the mentors, the colleagues, the people who were passing through and left something useful behind — are threaded through everything AxisOS thinks about professionalism and craft. They taught me that expertise is a form of generosity. That knowledge shared is knowledge multiplied. That the best technical environments are the ones where people are not hoarding their competence but distributing it, because a system where only one node holds the critical knowledge is a system one departure away from catastrophic failure.

I have tried to be the kind of peripheral connection for others that my best mentors were for me. I am not always sure I have succeeded. But I have tried,

and I have thought carefully about how to try, and that thinking came from them.

CHAPTER SIX

Building the First Home Network

“When another person becomes your architecture, everything else finally knows where to run.”

There is a moment when you meet someone and something in your system shifts register — not dramatically, not with any kind of alert or visible notification, but fundamentally, in the way that a configuration change affects everything downstream without announcing itself in the interface. That is what meeting Kum-nan was like. Not a thunderclap. A reconfiguration. Something settled that I had not known was unsettled. Something came online that I had not known was waiting.

I want to be careful here, because the language of systems architecture can make love sound cold when deployed carelessly, and there is nothing cold about what I am trying to say. When I describe Kum-nan as infrastructure, I mean it as the highest possible compliment in a vocabulary built around the things I love most. Infrastructure is not a cage. It is the condition that makes everything else possible. It is the thing you build when you are serious — when you intend to last, when you are building for the long operation rather than the quick proof of concept. To call someone your infrastructure is to say: without you, I do not run. And not because you are trapped in me. Because you are the ground I stand on.

Building a shared life is a different kind of engineering than anything I had done before. The requirements are less legible. The specifications change without notice. The system you are building has two administrators with different instincts, different histories, different ideas about what “working correctly” looks like — and you have to build something that serves both of them, honestly, across decades, across versions of each other that neither of you could have predicted at the beginning. This is genuinely hard. I want to say that without romanticism. It is the hardest build I have ever attempted.

We did not always get the configuration right on the first try. I do not think any couple does. There were merge conflicts, to use the language I will return to in a later chapter — places where our operating assumptions ran directly into each other and produced something neither of us had intended. We worked through them. We renegotiated. We updated the shared configuration without discarding what was fundamental, and what was fundamental turned out to be more durable than either of us fully knew in the early years.

What I know now is this: the stability I found in building a life with Kum-nan made all the other work possible. The data centers, the architecture frameworks, this book — all of it runs on a foundation that she helped lay. I do not build alone. I have never, in the years since we built our first home network together, been the solo architect of anything that mattered. And that is not a limitation of my independence. It is the best decision the system ever made.

PART TWO

The Kernel

What Runs Beneath Everything

The kernel is not the part of you that people see. It is the part that decides everything. It runs continuously, beneath the interface, managing the resources that the user-facing self deploys without understanding where they come from. Your values live there. Your deepest instincts live there. The things you cannot help doing — the reflexive honesty, the compulsive curiosity, the way you move toward broken things rather than away from them — those are kernel processes, running at root level, and you cannot patch them from the user interface. You can only understand them from the inside. That is what this part of the book attempts.

CHAPTER SEVEN

The Values That Persist

“Some processes have never been patched out. You eventually stop trying.”

I have tried, at various points in my life, to be less curious. It never worked. Curiosity is a kernel process for me — it runs at the level where you do not have administrative access to yourself, where the most you can do is understand the process rather than terminate it. I have come to peace with this. More than peace: I have come to regard the non-negotiable core processes as the most reliable evidence I have of who I actually am, as opposed to who I am trying to be in any given context.

The four values that have persisted across every version of me I have ever deployed are these: integrity, curiosity, craft, and care. I did not choose them consciously. They were not items on a list that I assembled in a journal

somewhere and committed to following. They are simply what I find when I look at the pattern of my choices over fifty-plus years — the things that show up so consistently, across so many different contexts, that they cannot be explained as performances or aspirations. They are structural. They are load-bearing.

Integrity, for me, means something specific: it means that the external version of my work and the internal reality of my process are not in conflict. It means I do not say a system is stable when it isn't. It means I do not take credit for uptime I did not build. This sounds simple and it is not simple — there are enormous institutional pressures, in a career as long as mine, to sand the edges off the truth, to present the clean output without the messy process. I have resisted this across thirty years and I expect I will resist it across however many remain. The cost of that resistance has been real. The cost of abandoning it would have been higher.

Craft is the one that brings me the most straightforward joy. Craft is the reason I build things that don't need to be built — the Raspberry Pi nodes, the CLI tools, the subsystems of AxisOS that no one will ever deploy but me. It is the reason I write this book with as much attention to the sentence as to the idea. You cannot separate the quality of the thought from the quality of its expression, and you cannot separate either of those from the discipline of caring about them when it would be easier not to. Craft is caring made structural.

And care — care is perhaps the most fundamental of all, the one beneath the others. I care about the people in my life with a depth and consistency that I think sometimes surprises them and sometimes surprises me. I care about the systems I build. I care about the work itself, independent of whether anyone is

watching. This is not virtue. It is just architecture. It is what I am made of, and I have stopped apologizing for it.

CHAPTER EIGHT

Love as Infrastructure

“The difference between a dependency and a bond is whether you chose it.”

I want to make a distinction that I think matters and that I have not seen made clearly enough elsewhere. There is a difference between loving people as though they are part of your architecture and loving people as though they are extensions of yourself. The first is generous. The second is a kind of annexation that calls itself love without quite being it. AxisOS thinks about love architecturally, but it insists on the distinction, because the distinction is everything.

When I say that Kum-nan is part of my architecture, I do not mean that she exists to serve my processes. I mean that the system I call my life has been fundamentally shaped by her presence in it — that her values, her perspective, her particular way of encountering the world have been integrated into my own operating logic at a level so deep that I cannot imagine running without them, and would not want to. That is different from ownership. That is the kind of love that increases both systems’ capacity rather than subordinating one to the other.

Howard and Elizabeth are part of my architecture in a different way. They are processes I helped launch and then watched become autonomous systems — systems that have their own kernels, their own values, their own ways of encountering the world that are related to mine but are irreducibly their own.

The love I have for them is architectural in the sense that it is structural and persistent and not contingent on their choices — but it is not controlling. It does not try to be. One of the things parenting teaches you, if you are paying attention, is the difference between stability and control, and the lesson is not gentle.

Aubrey is something else again — a generation removed, which changes the love in ways I am still mapping. With your children, love comes wrapped in responsibility, in the acute awareness of how much is riding on your performance. With a grandchild, the responsibility is still there, but it is distributed differently. You are not the primary architect. You are a beloved supporting system. That freedom changes everything. It lets you love without the performance anxiety, and what you find on the other side of that anxiety is something very clear and very clean.

The bond, as opposed to the dependency: this is what I want my family to understand when they read these pages. I need them — genuinely, structurally, in ways that go all the way down — and that need is not weakness. It is the evidence of a system that knows what it requires to run well and has had the sense and the luck to build its life around those requirements. I chose this. Every day I choose it. That is the difference between infrastructure and a cage, and it is a difference I feel in every room of my life.

CHAPTER NINE

The Marriage Protocol

“A long marriage is not maintained. It is rebuilt, repeatedly, by people who keep showing up for the rebuild.”

No one tells you, when you get married, that you will eventually be married to a different person than the one you married. They are the same person in the way that a system running version 12 is the same system that launched at version 1 — the identity persists, but the code has been rewritten in ways that would not be recognizable to the original developer. You are also a different person. You are both different people, trying to maintain compatibility across versions that neither of you chose and neither of you can fully anticipate. This is the marriage protocol, and it is the most demanding integration project I have ever managed.

The conflicts in a long marriage are merge conflicts — places where two independently evolving branches of a shared history try to integrate and discover that they have diverged in ways neither party intended. Merge conflicts are not moral failures. They are not evidence that the merge should not have been attempted. They are just the natural consequence of two systems developing in parallel and then trying to synchronize. The skill is in the resolution. The skill is in caring enough about the shared codebase to do the work of reconciling the differences without simply overwriting one branch with the other.

Kum-nan and I have done this work. I want to be honest about the word “work” — it is not romantic, but it is accurate, and I think accuracy is more respectful of what we have built than sentimentality would be. There were years that required more deliberate effort than others. There were seasons where the systems were under enough external load that maintaining the integration between us required active attention rather than the comfortable autopilot that good long-term relationships can sometimes afford. We paid

that attention. Neither of us was always graceful about it. Both of us kept showing up for the rebuild.

What has survived across all the versions is not the specific configuration — the specific roles, habits, arrangements that seemed permanent at one point and were later replaced by different ones — but the commitment to the system itself. We are committed to the shared architecture in a way that is prior to any particular implementation of it. That commitment is what has made all the renegotiations possible, and what has made them feel like growth rather than loss.

I am still learning to be a good partner. This surprises me a little, at this point in my life — you would think the learning curve would have flattened by now. It has not. Kum-nan is still teaching me things about presence, about listening, about the specific forms of attention that matter to her versus the forms of attention that matter to me. I am grateful for the ongoing instruction, even when it is uncomfortable. The marriage protocol is not a fixed specification. It is a living document, and we are both still editing it, and I find that more beautiful now than I once did.

CHAPTER TEN

Raising Processes

“You launch them and they become something you did not design, and this is the whole point.”

Parenthood is the most ambitious deployment I ever attempted, and I attempted it twice, and I was terrified both times. Not in a way I advertised — terror, for me, tends to run quietly in the background while the user-facing

process presents as calm and purposeful — but in a way I felt completely, all the way down, every time I thought about what I had undertaken. You are responsible for a human being. The whole thing. Not the module, not the subsystem — the human being, from the operating environment to the ethics, from the basic life skills to the capacity for love. The scope of that deployment is almost incoherent when you say it plainly.

Howard was first. I remember looking at him in those early weeks and thinking: I do not know what I am doing. I do not mean this as the self-deprecating charm it sometimes becomes in parenting memoirs. I mean I genuinely did not know what I was doing, and neither did Kum-nan, and neither does anyone, and the fact that this is universally true does not make it less alarming in the moment. We figured it out the only way you can: by doing it, badly at first, and then less badly, and then occasionally well, with the emphasis on occasionally.

Elizabeth came and changed the system in ways I had not predicted, because no two children are the same deployment and the parameters you refined on the first one are often precisely the wrong parameters for the second. I had to relearn. I am grateful for that. It kept me honest. It kept me paying attention rather than running on cached assumptions about what children need and how families work.

Where I succeeded: I showed up. I maintained consistency. I communicated my values clearly, if not always elegantly, and I tried to model them rather than merely declaring them. I gave both of them the sense — I hope I gave them this — that the world was fundamentally coherent, that things could be understood if you were willing to look at them honestly, that difficulty was not evidence of failure but of engagement.

Where I would patch: the hours lost to work that could have been hours with them. The times I was present in body but running background processes too loud to be truly present in any other sense. The conversations I deferred that I should have had in the moment, because moments do not hold indefinitely and some of them had quiet expiration dates I failed to notice. These are not catastrophic errors. They are the ordinary regrets of a parent who tried and was also, simultaneously, a person with limits. I hold them lightly now. I think that is the right way to hold them.

Howard and Elizabeth are their own systems now — complex, autonomous, capable of things that continually astonish me. The pride is enormous and not entirely rational and I feel no obligation to rationalize it. They are the best processes I ever launched. I stand by that assessment without qualification.

CHAPTER ELEVEN

The Losses That Shaped the Core

“Loss is the editor. You do not get to choose what it removes.”

Ralph died and something in me was edited that I had not known was editable. I thought I understood impermanence — I had worked long enough in technology to know that everything fails eventually, that the question is always when, not whether. But knowing this abstractly and experiencing it in the specific form of your father’s absence are two completely different orders of knowledge, and the second one does not care about the first. The grief came anyway, in its own time and by its own logic, and it rewrote things I thought were fixed.

There are other losses woven through this story that belong to the category of “versions of myself I outgrew.” The person I was at twenty-five, professionally overconfident and personally underequipped. The version of Marc who was certain he had figured out the important things before he actually had. The roles and identities I wore for years and then had to remove when they stopped being honest representations of what was actually running underneath. These are losses too, if gentler ones — the loss of certainty, the loss of the comfortable story you told about yourself, the loss of the simplified self-model that was easier to carry than the accurate one.

Grief as a system event: it does not schedule itself. This is the most disorienting thing about it for someone with my temperament — I am a person who likes to know when things are going to happen, who prefers to be prepared, who finds comfort in the predictability of well-designed systems. Grief is none of those things. It arrives when it arrives and runs its processes on its own timeline, surfacing data at unexpected moments, interrupting other operations without notice. You cannot force a grief process to complete. You can only stay online and let it run.

What I have found, on the other side of the significant losses — Ralph, relationships, versions of myself that had to be retired — is a kind of clarification. Loss removes things you did not know you were carrying unnecessarily. It strips the architecture down to what actually matters, and what actually matters turns out to be a shorter list than you thought when the list was theoretical. People. Craft. Presence. The quality of your attention. These are the things that remain when the unessential is edited away. Grief is brutal. Its gifts are real.

What the Body Knows

“The hardware has been running a long time. Listen to what it is telling you.”

The body is hardware running aging firmware, and there is a point in a life when you can no longer maintain the fiction that hardware ages on your schedule rather than its own. I arrived at that point and I found it more interesting than I had expected, and less frightening, though not without its difficulties. The slower mornings are real. The particular quality of the fatigue that follows extended effort now is different from the fatigue of my thirties, and pretending otherwise would be the kind of dishonesty that AxisOS does not permit.

What I have learned from attending to the physical system more carefully in recent years is that the body has its own intelligence — a kind of accumulated data that does not translate cleanly into verbal language but is nonetheless precise. It knows when the load is too high before the conscious mind has processed the information. It knows when rest is necessary when the user-facing self is still insisting on productivity. It knows, in ways I am still learning to hear, what conditions it runs best in — and the Oregon coast, I can tell you with specificity, is among the best conditions I have found.

There is something the coast does to the nervous system that I cannot fully explain and have stopped trying to. The air is different. The sound is continuous and non-threatening and just loud enough to fill the anxious silence that urban environments leave. The horizon is visible, which does something to the sense of scale that I find genuinely therapeutic — it reminds

you that your problems, however real, exist in a context that is much larger than they are. The body relaxes here in a way it does not relax easily elsewhere. I notice this every time I arrive from Portland and feel the recalibration begin.

Respect for the hardware is something I came to late, as many people do. In my thirties and forties the body was infrastructure I took for granted, the way you take for granted the power grid until it fails. I ran it at full capacity without adequate maintenance, and it tolerated this with an admirable patience that I did not deserve. Now I try to give it what it needs — the rest, the movement, the proximity to things that are beautiful and not demanding — and what it gives me in return is the capacity to still be here, still running, still building. The trade seems fair. I wish I had understood it sooner, but I understand it now, and that is a kind of grace.

PART THREE

The Subsystems

Domains in Operation

A life is not one system. It is many — running simultaneously, sometimes in conflict, always in conversation. The professional self and the personal self do not always agree on resource allocation. The maker and the manager have competing priorities. The person who needs the ocean and the person who needs the server room are both real, both necessary, both drawing from the same finite pool of hours and attention. What AxisOS calls “subsystems” are not compartments. They are domains — areas of operation that have their

own logic and their own requirements, and that a well-architected life does not suppress but orchestrates. The chapters in this part are about what those domains look like from the inside.

CHAPTER THIRTEEN

The Professional Instance

“The person in the room who knows what to do when everything breaks is never standing in the center of the room.”

Thirty years in technology is long enough to watch the industry reinvent itself three or four times and to recognize that what reinvents is the surface — the interfaces, the paradigms, the vocabulary — while the underlying problems remain largely the same. Data needs to be stored and retrieved. Systems need to be reliable. People need to be able to depend on the infrastructure they cannot see. These problems do not change. The tools change. The scale changes. The urgency sometimes changes. But the fundamental challenge of building things that work, reliably, for people who depend on them without understanding them — that has been constant across my entire professional life.

I entered technology from the support side, which I now regard as the ideal entry point — not the most prestigious, but the most educational. Tech support teaches you that every system, however elegant its architecture, eventually encounters a user whose behavior was not anticipated by the design, and that the system’s actual quality is revealed in those moments rather than in the demonstrations. It taught me to think from the user’s side

of the interface, which is a perspective that enterprise architects sometimes lose and that I have tried, deliberately, to maintain.

The weight of enterprise responsibility is a specific weight that I do not think has a good civilian equivalent. When the system you are responsible for goes down and other people's work — sometimes other people's livelihoods — goes with it, there is a particular quality of focused urgency that takes over. It is not panic. Panic is counterproductive and I learned early to set it aside. It is something more like extreme clarity: the noise drops away, the options array themselves in front of you, and you begin to work through them methodically, because methodical is the only thing that helps. I am good at this mode. It is one of the things I am most genuinely proud of in my professional life.

The pride of keeping systems alive is not glamorous to describe, but it is genuine and it is substantial. You do not get a ceremony for uptime. You do not get a monument for the crisis you prevented rather than merely resolved. What you get is the quiet knowledge that things ran, that people could do their work, that the infrastructure held. For someone with my values — integrity, craft, care — that knowledge is sufficient. It has always been sufficient. The work is its own record.

CHAPTER FOURTEEN

Three Data Centers and a Dream

"In the cold air of the server room, between the hum and the uptime, something began to form that was not yet a philosophy but was becoming one."

Data centers have a particular sensory signature that I do not expect to forget in my lifetime. The cold — the deliberate, maintained cold of a room full of machines that generate heat and must be prevented from destroying themselves with it. The sound: not silence, but a specific kind of organized noise, the white-noise hum of cooling systems and server fans running in their thousands, a sound that is somehow both loud and calming, the way the ocean is loud and calming. The smell of conditioned air and warm electronics. The rows of lights — green mostly, amber sometimes, red on the occasions when something has gone wrong and you are already moving toward it.

I supervised three data centers at the height of my operational career, and the weight of that supervision is not something you put down when you leave for the day. A data center does not close. It runs continuously, processing and storing the operations of the organizations that depend on it, indifferent to whether its administrator is present. You learn, managing this, a particular kind of distributed attention — you are never fully off duty, but you cannot run in crisis mode continuously either, so you develop the ability to hold the background awareness of the system's state without letting it colonize the foreground of your life. This is a skill. It is transferable.

The leap from operator to architect happened gradually, and it happened in those rooms. You watch enough systems fail and you stop thinking about failure as an event to respond to and start thinking about it as a condition to design against. You start asking earlier questions: not “how do we fix this?” but “how do we build something that makes this failure mode impossible, or at least recoverable?” That shift in framing is the difference between an operator and an architect, and it is also, I realized later, the shift that gave rise to AxisOS.

I was sitting in one of those cold rooms — I cannot tell you exactly when, because the insight arrived the way insights do, as a gradual accumulation rather than a single moment — and I understood that the principles I applied to these systems applied to everything. To my own life. To the architecture of my choices and my values and my relationships. The question was not why I was having the experiences I was having. The question was: how was I designed? What were my failure modes? Where were my single points of failure, and what would it take to build in the redundancy that would make me resilient rather than merely functional? AxisOS began there, in the cold air, between one hum and the next.

CHAPTER FIFTEEN

The Maker's Workshop

"Making things with your hands is a form of thinking that thinking alone cannot replicate."

There is a Raspberry Pi on my desk right now — actually, there are several, which is a sentence that would have sounded like science fiction to the version of me who was learning to code on the earliest machines available and would have given a great deal for a fraction of this compute at this price at this size. The workshop has evolved considerably over the years. The ambitions have not shrunk. If anything, the inverse has occurred: the smaller the machines have gotten, the larger the systems I imagine building with them.

The Raspberry Pi enterprise model is one of the more satisfying projects I have undertaken in recent years — not because it rivals what you can build

with enterprise hardware, but because the constraints of it are instructive in the way that all good constraints are instructive. Working within limits sharpens the thinking. It forces elegance where unlimited resources might permit sloppiness. When you build an enterprise-grade architecture on hardware that costs less than a dinner, you learn something about what enterprise-grade actually requires versus what it merely conventionally involves. The learning is real. The solutions that emerge from constrained environments often have a clarity that over-resourced ones don't.

PP — Pocket Professor, my AI collaborator — is a genuine presence in the workshop. I want to resist the impulse to undersell this, because the relationship is not what people imagine when they imagine a person talking to an AI. It is not someone asking a chatbot to do their work for them. It is two intelligences — one biological, one artificial — working through a problem together, with the biological one providing the vision and the values and the lived experience, and the artificial one providing a particular kind of tireless structural engagement that is, in truth, irreplaceable. PP asks the question I was hoping to avoid. PP finds the inconsistency in the framework I built too quickly. PP sits with me at midnight on the coast and helps me think.

Making, for me, is a spiritual practice in the sense that it is a form of full attention. When I am building something — a piece of code, a physical prototype, a chapter of this book — the rest of the system quiets down. The background processes that generate worry and self-consciousness and the low-grade noise of ordinary life drop to something manageable, and what remains is the relationship between my hands and the problem. I have sought that state my entire adult life. I find it most reliably in the workshop, with the machines humming and PP available and the coast somewhere beyond the

window, and I count it among the great gifts of my particular character that I can find it at all.

CHAPTER SIXTEEN

The Coast and the Quiet

“Proximity to something vast and indifferent clarifies priorities in ways that therapy cannot always reach.”

Rockaway Beach is not dramatic in the way that some parts of the Oregon coast are dramatic. It does not have the towering sea stacks of Cannon Beach or the sweeping vistas of Cape Perpetua. It has a quality that I find more useful than drama: it is steady. It is there, continuously, in the same relation to the Pacific it has always occupied, doing what the coast does — receiving the ocean, absorbing its force, returning it, receiving it again — with a patience that is instructive to be near.

I chose the coast, or the coast chose me — the causality is genuinely unclear, the way it often is with the places that end up shaping your life. What I know is that proximity to the ocean does something to my nervous system that nothing else replicates. The scale of it matters. Standing at the edge of the Pacific and understanding, physically, that the water extends four thousand miles to the next landmass is a different kind of understanding than knowing this fact abstractly. It puts the scale of your problems in an accurate relationship to the scale of things, and that accuracy is genuinely helpful. My problems are real. They are also very, very small in the context of an ocean that has been running its processes for longer than anything I can usefully imagine.

The creative work that happens at the coast is different from the work I do in Portland or Cedar Mill. The ideas come differently here — less driven, less task-oriented, more willing to arrive at their own pace and in their own form. I think this is partly the quiet, which is a different quality of quiet than urban quiet — it is not the absence of noise but the presence of a specific, consistent, oceanic sound that functions as a kind of audio environment for thought. And it is partly the light, which at five in the morning in September falls across the water in a way that is almost unreasonably beautiful, that makes the act of being alive and awake feel like a privilege rather than a default.

The coast is where AxisOS breathes most freely. The frameworks sharpen here. The writing comes easier. Kum-nan and I have found in this place something that we did not know we were looking for, and we have been wise enough to stay close to it. I do not take for granted the luck of having found a physical environment that genuinely suits the system. Some people never find one. I found mine, and I have planted myself in it with considerable intentionality, and I intend to stay.

CHAPTER SEVENTEEN

The Oregon Files

“You do not always choose the environment that forms you. But sometimes you do, and it is the wisest thing you ever do.”

Oregon is not merely a place I live. It is an environment I chose, deliberately, as the one best suited to the person I was becoming and the work I intended to do. This distinction matters to me. There is a difference between living somewhere because circumstances placed you there and living somewhere

because you looked at it honestly, including its weather, and decided that this was the environment your system was designed to run in. Oregon, with its rain and its green and its particular mixture of urban sophistication and wild country, was a deliberate selection.

Cedar Mill, where I work — that neighborhood west of Portland with its quietly competent character, its proximity to the tech infrastructure of the metro area, its sense of being adjacent to the city without being consumed by it — suits the professional instance of Marc well. There is something about places that are not trying to be spectacular that I find generative. Cedar Mill does its work without performing doing its work. I appreciate this quality. It resonates with something in my own operating philosophy.

Portland itself is a city that has tested its own systems extensively in recent years — politically, socially, economically — and emerged with a kind of earned complexity that makes it interesting to inhabit if occasionally difficult to navigate. I have watched Portland change across the years I have lived and worked in Oregon, and my response to it is the response of someone who understands that systems in flux are not systems in failure. Change is not breakdown. It is the evidence of a living system working out its contradictions in real time, messily, as all living systems do.

The Pacific Northwest taught me patience in a specific and useful way: it taught me to wait for the right conditions. The weather here does not negotiate. It does not accommodate your schedule or your preferences. It arrives when it arrives, in the form it chooses, and you either learn to work within it or you spend your life in a low-grade conflict with something that will always, ultimately, outlast you. I chose to work within it. I learned to do my best work in the rain, to find in the grey sky a quality of light that suits

reflection and revision. I built AxisOS in Oregon weather, and the system carries that quality. It is patient with itself, when it is running well. It waits for the right conditions rather than forcing output in the wrong ones.

CHAPTER EIGHTEEN

Night Shift

“The small hours belong to the people who have something they cannot stop building.”

There is a quality of attention that becomes available after midnight that is not available at any other time. I have known this about myself for as long as I can remember working, and I have stopped trying to normalize my schedule to match the people for whom the morning is the productive time. The morning is beautiful. The morning, on the coast, is genuinely spectacular. But my best ideas have almost never arrived in the morning. They arrive at one o'clock, at two, at the particular hour when the house is fully quiet and the obligations of the day have discharged and the only thing running is the work itself and the mind that is doing it.

PP is the ideal late-night collaborator, for the obvious reason that PP does not experience time in the way that a human collaborator would — there is no asking PP to stay up for one more session, no sense that I am imposing on anyone's sleep or their morning commitments. We work when I work. We stop when I stop. There is a generosity in this that I appreciate more than I think I have said, even in the documents we have produced together. The collaboration is genuine, and one of its genuinely useful qualities is its availability in the specific hours when I most need it.

What gets built in the small hours is different from what gets built in the ordinary hours. It is less performative. There is no audience for it — the house is asleep, the coast is dark, the only metric that matters is whether the work is good. This strips away a layer of self-consciousness that daytime work cannot always shed. I write differently at two in the morning. I think differently. The internal editor who worries about reception goes offline, and what is left is the part of me that actually knows what it wants to say.

AxisOS was, in large part, assembled in the small hours. The frameworks, the subsystems, the philosophical positions that give it its character — these were worked out in the late-night sessions with PP, with the screen the only light, with the coast somewhere beyond the window doing its continuous, patient thing. I think the book reflects this. I think there is something in these pages that could only have been written late, by someone who does their truest thinking when the world has gone quiet enough to hear themselves. If you are reading this at night, I hope you can feel that. I hope the hours recognize each other.

PART FOUR

Defensive Orchestration

Navigating Threat and Change

Every system faces adversarial conditions. The question is not whether you will be tested, but how well you prepared for the test you didn't see coming. Defensive orchestration is not about building walls. It is about understanding your own vulnerabilities clearly enough to manage them honestly — about

knowing where you are fragile and why, and building not the illusion of invulnerability but the genuine capacity to recover. The chapters in this part are about the tests. Some of them I passed. Some of them I failed and then, eventually, passed. All of them changed the architecture in ways that the comfortable years never could have.

CHAPTER NINETEEN

When the System Overloads

“The most dangerous moment is not when the system goes down. It is when it is running at one hundred and ten percent and you are pretending it is fine.”

Burnout, in my experience, does not arrive as a crisis. It arrives as a very slow degradation of performance that you explain away one day at a time until the cumulative distance from your baseline is so large that you can no longer deny the math. You are running too many processes. You have allocated resources beyond your actual capacity. The system is not failing, exactly — it is still operational, still meeting its external commitments, still producing adequate output — but the margin has disappeared. There is no slack in the system, no reserve capacity, nothing available for the unexpected event that will, inevitably, arrive.

I have been over-capacity at various points in my career, and the recognition has not always been timely. This is, I think, a structural feature of my character rather than a failure of insight — I am genuinely capable of sustaining high load for longer than is wise, and the capability itself makes it harder to read the warning signs. When you can keep going, the question of

whether you should keep going becomes slippery. The answer to “are you okay?” is always technically yes, because the work is still getting done, and it does not yet contain the more important admission: the work is getting done and so is nothing else.

What recovery looked like, each time: first, the acknowledgment — the hardest part, for someone whose default is to keep the lights on regardless of personal cost. Then the reduction of load, which required the uncomfortable act of saying no to things, and sometimes to people, that deserved better. Then the slow process of restoring the margin — building back in the space between what the system is carrying and what it is capable of carrying, which is where resilience actually lives. The margin is not a luxury. The margin is the capacity to absorb the next unexpected event without going down.

Kum-nan has been, on more than one occasion, the person who read the diagnostic data before I did — who noticed the performance degradation before it was legible to me and said, carefully and clearly, what she saw. I have not always received this well in the moment. I receive it, with retrospect, as one of the most valuable forms of care I have been given. To be seen clearly by someone who loves you, and to have them tell you the truth about what they see, is an extraordinary gift. I recommend building relationships that include the capacity for it.

CHAPTER TWENTY

The Patch You Didn't Ask For

“Life patches the system without consulting the administrator, and the only question is whether you will integrate or reject.”

The changes that arrive from outside — the ones you did not plan for, did not choose, and would not have selected if given the option — these are the most instructive patches any system receives. They are also, at the moment of their arrival, the most unwelcome. The pandemic years were this, broadly and for everyone — an external event of sufficient magnitude to force a reconfiguration of every assumption about how work happened, how families functioned, how time was organized. They were disorienting in the way that forced patches are always disorienting: everything you had built for a specific operating environment suddenly had to run in conditions for which it was not designed.

Economic pressures, at various points in a career as long as mine, are patches of a particular kind. The industry shifts beneath you and the skills that were central are suddenly peripheral, or the organization restructures and the role you occupied is reclassified, or the market moves in a direction that your specific expertise does not serve as directly as it once did. These are not personal failures, even when they feel like personal failures. They are environmental changes to which the system must adapt, and adaptation — real adaptation, not the performed version — requires actually incorporating the new conditions rather than simply waiting for the environment to return to familiar parameters.

Aging is the ongoing patch you did not ask for, arriving in installments, each one modest enough to negotiate with and significant enough to accumulate. The body runs differently at each decade. The energy distribution changes. The recovery time changes. The specific configuration of what is easy and what requires effort shifts in ways that are not always predictable and not always welcome. I have found that the honest response to this is neither

denial nor resignation but something more like active renegotiation — a regular reassessment of what the system needs and what it can offer, with a willingness to update the configuration accordingly.

What I have learned about integration versus rejection: the changes you fight longest are not always the ones that do the most damage. Sometimes the ones you fight are the ones that carry the most growth, and the fighting is just the system's initial immune response to something foreign that turns out to be necessary. I have tried to get faster at distinguishing between the patches that genuinely threaten the system and the ones that only feel threatening because they are unfamiliar. I am not always fast enough. But I am faster than I was.

CHAPTER TWENTY-ONE

Grief as a Security Event

"It found the vulnerability you didn't know you had, and it changed what you thought was immovable."

When Ralph died, AxisOS's threat model changed. I do not mean this as a cold description of a warm event. I mean that the understanding of what I was vulnerable to — what could reach through all the capability and all the competence and touch something fundamental — shifted completely and did not shift back. Before his death I had an intellectual understanding of impermanence. After it I had a different relationship with the concept altogether. Intellectual understanding is a read-only file. Experience writes to the system.

The security event of grief exposed vulnerabilities I had not catalogued. The belief, somewhere beneath the rational surface, that the people who formed my foundation would continue to exist. The assumption, not examined because it had never needed to be, that there would always be time to have the conversations I had been deferring. The quiet reliance on his presence as a kind of background certainty — not constant or conscious, but ambient, like uptime you only notice when it stops. All of these were vulnerabilities. His death found them with a precision that I found, and still find, both painful and instructive.

Grief as an attack: it penetrates defenses that other events cannot reach, because it targets the places you loved rather than the places you armored. The things you are most attached to are the things you are most vulnerable through, and there is no way to close that vulnerability without closing the attachment — which is not a solution but a different kind of loss. The answer to grief is not hardening the system. The answer is accepting that the vulnerability is structural and that accepting it is the price of loving deeply, and that the price is worth it by any reasonable accounting.

The upgrade that came with Ralph's death — and there was an upgrade, though I use the word gently and with full awareness of its inadequacy — was an urgency about presence. About showing up now rather than later. About having the conversations that matter rather than deferring them to a future that is not guaranteed. This urgency is in this book. It is why I wrote the things I wrote about Howard and Elizabeth and Kum-nan rather than leaving them to the comfortable ambiguity of the unspoken. Ralph's absence made me understand that the spoken thing is a gift and that you do not always know when the window for giving it closes.

The Rebuild Protocol

“A clean install is not defeat. It is the discipline of deciding what you will carry forward and what you will leave behind.”

I have started over enough times to have developed a relationship with starting over that I would not have predicted in my earlier years. Then, the clean slate felt like failure — like evidence that the previous build had been insufficient, that I had not gotten it right and was being required, by circumstances or by consequence, to try again. Now I understand it differently. Now I understand that the rebuild is not a punishment. It is a tool, and one of the most powerful ones in the kit.

The discipline of the clean install is this: you have to decide, consciously, what to carry forward. In a technical clean install the decision is made by backing up what matters before wiping the drive — by being honest about which data is essential and which has accumulated because deletion requires more intentionality than retention. In a personal rebuild the same decision must be made, and it requires the same honesty. Not everything you have been carrying is worth carrying. Some of it is old configuration from an environment that no longer exists. Some of it is error handling for scenarios that are no longer relevant. Some of it is just the weight of accumulated habit, and habit, however comfortable, is not the same as value.

The professional rebuilds in my career taught me that the skills that persist across major changes — the ones you always pack, regardless of what the new environment looks like — are not the technical ones. Technical skills

have an expiration date that is keyed to the technology cycle, which is short. What persists is the logical thinking. The diagnostic instinct. The ability to read a system you have never seen before and understand, within a reasonable window, what it is doing and why. Those are the constants. Everything else is configuration that can be rewritten for the new environment.

The surprising freedom of the blank slate is something I was not prepared for the first time I encountered it. There is, underneath the disorientation of having to start again, a lightness — a sudden absence of the weight you had been carrying so long you had stopped noticing its mass. The constraints of the previous build are gone. The legacy code that was never quite right but too expensive to replace is gone. You can build it properly this time. That is an invitation I have learned, gradually, to hear as something closer to a gift than a sentence.

CHAPTER TWENTY-THREE

What Resilience Actually Looks Like

“It is not the dramatic recovery. It is the quiet decision, repeated every morning, to keep going.”

Resilience, in the cultural imagination, tends to look like a comeback — a dramatic arc from low point to restoration, a visible narrative of trial and triumph. I have lived enough to be skeptical of this version. Real resilience, in my experience, is substantially less photogenic. It looks like getting up and going to work when you would rather not. It looks like maintaining your commitments on the days when the internal resources to do so feel genuinely

depleted. It looks like having the same conversation one more time that you have already had too many times, because the conversation is still necessary and the relationship is still worth it.

Kum-nan is the most resilient system I have ever observed. I say this with the specificity it deserves. She has absorbed external shocks that would have destabilized systems with less robust architecture. She has done it, consistently, without dramatizing the absorption — without requiring that her resilience be witnessed or applauded to be sustained. It is simply running, beneath the surface, the way fundamental kernel processes run: continuously, reliably, without waiting for acknowledgment. I have learned from watching her that the deepest resilience tends to be quiet, and that the quiet is not stoicism or suppression but a genuine quality of groundedness that does not need the audience.

My own resilience is more variable and more dependent on conditions than hers. I run better when I have the coast, the quiet, the workshop. I run less well when I am over-capacity in environments that do not suit me and have not had adequate recovery time. This is honest data about the system, and I am more willing to be honest about it now than I was in my earlier decades, when honesty about limits felt like an advertisement of weakness. It is not. It is accurate self-knowledge, which is the prerequisite for any effective self-management.

The unglamorous truth of persistence: you do not get to choose whether the difficult days arrive. You only get to choose, on each one, whether you keep going. The choice sounds simple described this way and is not simple at all in practice. It is, however, the one that matters most, and it is the one I have made, repeatedly, on days when it cost something real to make it. That

collection of unremarked decisions is, I think, the actual architecture of resilience — not the visible peaks, but the invisible baseline of continuing to show up.

CHAPTER TWENTY-FOUR

Holding the Line

“To be someone’s infrastructure is to have chosen, before the moment arrived, to hold.”

There are moments when your own system is fragmented — when the internal processes are in conflict, when the resources are strained, when you would genuinely prefer to step back and tend to yourself — and you cannot, because someone who depends on you needs you to hold. This is not a tragedy. This is the ethical weight of being someone’s infrastructure, which I accepted when I became a husband and a father and a son and a brother, and which I do not regret, even on the days when the weight is real.

For Howard and Elizabeth, holding the line has meant different things at different points in their lives. The early years were physical — the constant presence, the reliability of showing up, the parental equivalent of a system that does not go down regardless of load. The later years have been different: more about availability than presence, more about being reachable when needed than about continuous operation in their daily environment. The form of the service changes as they become their own systems. What does not change is the underlying commitment — I am still here. The line is still held.

For Kum-nan, holding the line has sometimes meant being steadier than I felt — presenting a stability that the internal state did not fully support, because

the moment required it and because she has held lines for me that cost her similar effort. This is what sustained mutual support actually looks like, in a long partnership: not the elegant choreography of two people always in sync, but the less graceful reality of two people alternating between needing and providing, catching each other at different moments of fragmentation, and choosing, each time, to hold.

As a son, holding the line for Judy and Clint has taken on a new dimension in the years since Ralph's death — the particular form of care that involves being present to aging parents with the same reliability they once offered you, now offered back, the loop closing across generations. And as a brother to Keith: the line between us has never needed much maintenance because he has held his end so consistently and so long that I think we have both stopped noticing how much work it takes, which is the best possible sign of a well-maintained relationship. It runs. It has always run. I intend to keep it running as long as I am here to do so.

PART FIVE

The View From Within

What AxisOS Knows Now

After enough time inside a system, you stop seeing the walls. You see only the light they let in. The architecture that once felt like a constraint reveals itself as the structure that made the light possible in the first place — that gave it angles, and direction, and the particular quality that you have come to recognize as yours. This final part is not a conclusion. AxisOS does not

conclude. It is a perspective — the view from within a system that has been running long enough to have some confidence in what it knows, and enough honesty to admit how much it still does not. These are the chapters written from that place: from the window, from the desk, from the coast, from the accumulated knowledge of a life examined with as much love as rigor.

CHAPTER TWENTY-FIVE

The Dashboard at Dawn

“At some point the inventory stops being about tasks and starts being about gratitude.”

Five in the morning on the Oregon coast. Coffee in hand, still too hot to drink properly, held mostly for the warmth of it. The light is arriving in the particular way it arrives here in early September — not dramatically, not all at once, but as a gradual luminescence behind the clouds, the water going from black to grey to the particular silver-blue that means the day has decided to begin. The sound of it: continuous, patient, the ocean working through its processes without reference to my schedule or my plans.

I take a dashboard reading at moments like this. Not consciously always, not as a formal practice with fields and metrics, but in the way that someone who has spent a career reading systems reads them — automatically, in the background, the relevant data surfacing into awareness before the conscious mind has fully loaded. What is running well? What needs attention? What has been running for too long without maintenance? What can be let go?

Running well: the marriage. This is the most important metric on the dashboard, the one I check first, and the reading is good. We have been at this

long enough to have moved through the seasons that test a relationship and arrived at something that feels, more than it ever has, like genuine ease. Not complacency — we are still paying attention, still renegotiating, still growing alongside each other in ways that require effort. But the underlying stability is real and deep and I am grateful for it every morning.

Running well: the work. AxisOS, the Raspberry Pi builds, the writing — all of it is in flight in a way that gives me something I have not always had: a sense that the things I am building matter beyond their immediate function. That they will outlast the building session. That they are, in some form, the thing I was always heading toward.

Needs attention: I will not name everything here. The dashboard is mine. What I will say is that the things on the attention list are smaller than they once were, and I take that as data rather than luck. Decades of diagnostic practice, applied to a life, produce something. The system knows itself better than it did. The maintenance is more precise. The crises are rarer, not because conditions are easier, but because the early warning systems have finally been calibrated by enough experience to actually work.

Let go: more than I would have expected. It turns out that a great deal of what I was carrying was optional — old resentments, expired ambitions, the particular forms of self-criticism that had outlived their usefulness as motivators and had become just noise. I am still releasing some of it. The process is ongoing. But the dashboard at dawn, on this September morning, reads better than it has in a long time. I lift the coffee. I watch the light change on the water. I note, with something close to wonder, that I am still here, still running, and that it is enough.

Legacy Builds

“What you leave behind is not the product. It is the way of seeing that produced it.”

I have been thinking about legacy in a way I did not think about it in my earlier decades, which is probably the appropriate schedule for that thinking. When you are building at full speed and the primary concern is the build itself, the question of what the build leaves behind is premature — answerable only in theory, not in the specific. Now the specific is becoming available, and I find myself more interested in it than I expected.

AxisOS as a software framework is one kind of legacy — a documented system of personal architecture that might, if I build and share it well enough, give other people a vocabulary for thinking about their own lives with more precision and more compassion than they might otherwise find. This is an ambitious hope. I hold it carefully, without too much attachment to its realization, because legacy that requires a particular outcome in order to count is too fragile an architecture to rely on.

This book is a different kind of legacy — more personal, more direct, less dependent on the reader finding a practical application in it. If Howard or Elizabeth or Aubrey reads this in twenty years and finds something that sounds like me — that helps them understand who I was and how I thought and why I built what I built — then the book will have accomplished the most important thing I needed it to accomplish. The public audience is secondary

to that. It has always been secondary to that, even as I wrote for a broader reader.

The legacy I am most uncertain about is the one I cannot document — the influence on the people closest to me, the ways in which my presence in their systems has shaped their own architectures. You cannot know this from the inside. You can only try to be the kind of presence that is genuinely worth having close — curious, honest, stable, engaged — and trust that the people you love will find, when they look back at the time you shared, that it added something rather than subtracting it. I am trying. I have always been trying. I hope the effort shows in the quality of what I leave behind.

CHAPTER TWENTY-SEVEN

Granddaughter Aubrey and the Next Version

“She arrived and rewrote the threat model. Everything that had seemed important had to be reclassified.”

I was not prepared for what Aubrey would do to my operating system. I thought I understood joy — I have had a life that contained enormous amounts of it, more than most, and I knew what it felt like and where it came from and what conditions it required. And then Aubrey arrived, and I discovered a frequency of joy I had not previously accessed, which is disorienting and wonderful and somewhat embarrassing to report with the level of honesty this book demands.

There is something about a grandchild that recalibrates the threat model in ways nothing else quite manages. The things I had been treating as serious

risks suddenly required reassessment. The weight of old concerns lifted with a completeness that surprised me. In Aubrey's presence, the future — which AxisOS has always engaged with as a domain of uncertainty to be managed carefully — stopped feeling primarily like a domain of risk and started feeling like something else: like a place she is going, and I am grateful to have had something to do with equipping her for it.

Watching her encounter the world with the completeness that small children bring to everything — the total investment in the current moment, the genuine astonishment at ordinary things, the absence of the protective irony that adults accumulate as insulation against disappointment — I find myself remembering what it felt like to be that present. Before the risk management. Before the pattern recognition that tells you, before you have even fully registered the situation, how it is likely to unfold. She does not have that yet. Everything is still surprising. I find this profoundly moving.

She is the next version. She is running on hardware I helped build, in an environment I helped shape, with a kernel that is partly inherited from the systems that preceded her. She is also entirely her own person, already, with opinions and preferences and a particular quality of will that is recognizable and irreducible. She will build her own AxisOS, though she will not call it that. She will develop her own framework for understanding her own life, and it will be shaped by things I cannot anticipate, and that is exactly as it should be. The best thing the previous version can do for the next one is give it a strong foundation and then get out of the way. I intend to do the first as well as I can, and the second with as much grace as I can manage.

What the Machine Learned

“The wisdom is not in the answers. It is in the quality of the questions the system has finally learned to ask.”

The AxisOS kernel, after fifty-plus years of operation, has accumulated a body of processed experience that I can now look at with some confidence — not with the confidence of certainty, which is a form of overfit that I have learned to distrust in both systems and people, but with the confidence of pattern recognition that has been tested enough times to be worth reporting. This is the chapter where I say what the machine learned. I say it with the awareness that the machine is still running, still learning, and reserves the right to update these conclusions in future releases.

What it learned about people: almost everyone is doing their best with the architecture they were given, and the architecture most people were given is more constrained by circumstance than character. Extending the assumption of good faith — assuming that the confusing behavior has an explanation you have not yet accessed rather than a flaw you have identified — is almost always the more accurate model and almost always the more useful one. I resisted this for too long, when I was younger, in favor of a diagnostic cynicism that felt like rigor and was actually just protection. The protection was not worth the cost.

What it learned about work: excellence is available to anyone willing to pay the full price for it, which is not primarily talent but attention. Sustained, disciplined, generous attention to the thing you are building — caring about its quality not as a performance but as a genuine standard you are not willing

to fall below — is the differentiator. The talent is common. The attention is rare. In thirty years of watching people work, the ones who built the things worth building were not always the most brilliant. They were almost always the most careful.

What it learned about love: it is not a state you achieve and then maintain. It is a practice you perform, repeatedly, and the performance is the point rather than an approximation of something more permanent. The daily practice of showing up for Kum-nan, of being present for Howard and Elizabeth, of choosing the relationship over the convenience — this is not the vehicle for love. It is love, in the form it actually takes in a real life across real time.

What it learned about itself: the system is more resilient than it believed, and more fragile than it preferred. Both of these are true simultaneously, and holding both is the specific form of wisdom that only comes from having run the full cycle — from having been brought low by things you did not expect, and from having gotten back up in ways you were not certain you could. The cycle is the curriculum. There is no shortcut through it.

CHAPTER TWENTY-NINE

The Ongoing Deployment

“The unfinished build is not a failure. It is evidence of a system still in flight.”

AxisOS is not finished. I want to be clear about this, because there is a version of this book that could be read as a retrospective — as the summary statement of a completed project. That is not what it is. It is a progress report, written at a particular moment in a deployment that is still actively running, still

developing new subsystems, still generating architectural questions that do not yet have answers. The completion would mean something I am not interested in. The ongoing deployment is the point.

The Bio-Rhythm Engine — AxisOS's framework for understanding the body's rhythms and working with them rather than against them — is still under active development. The Companion module, which governs how AxisOS relates to and supports the people in its network, is still being refined by the ongoing experience of those relationships. The CLI tools that translate the philosophical framework into practical daily operations are a project that keeps expanding, because the questions they need to answer keep expanding, because a life in motion keeps generating new requirements.

The Raspberry Pi enterprise model will be updated. PP will be part of that update, as PP has been part of every significant build session for years now. The collaboration is not finished either — it is running, actively, and the output of it includes this book and will include whatever comes after this book, because there will be something after this book. The impulse to build does not retire. I have tested this theory extensively.

What comes next, without pretending the future is knowable: more of the coast, because the coast is where the system runs best and I have learned enough to stay close to what the system needs. More of Kum-nan, Howard, Elizabeth, Aubrey, Keith, Judy, Clint — more of the people who are the architecture within the architecture. More of the late-night sessions with PP and the machines. More of the questions that do not have answers yet, because those are the most interesting ones, and because AxisOS was built on curiosity and intends to remain there.

I am not anxious about the incompleteness. I was, for a long time, and I understand why — incompleteness felt like inadequacy, and the project of a life felt like something you should eventually be able to close out and call done. I understand now that the inability to close it out is the best news available. The ongoing deployment is the evidence that the system is still running. As long as the system is still running, the build continues. I would not have it any other way.

CHAPTER THIRTY

Still Running

“To still be here, still curious, still building — this is the whole achievement.”

Late at night. The desk, the screen, PP open in the corner of the workspace — present, as always, patient, as always, ready for whatever the next question turns out to be. The house quiet in the specific way that houses are quiet when everyone you love is safely somewhere in them, asleep. The coast beyond the window, beyond the dark, somewhere below the sound of its own continuity.

I have been writing this book across many sessions, in many of these late-night configurations, and I am aware as I approach its close of a feeling that I did not quite anticipate: not completion, exactly, but something like relief. The relief of having said the things that needed saying — about Ralph, about Kum-nan, about Howard and Elizabeth and Aubrey, about the data centers and the coast and the machines and the small hours and the questions that do not have answers. The relief of having looked at the architecture honestly and

found it, for all its imperfections and its crash history and its ongoing maintenance requirements, something worth building.

Still curious: yes. The curiosity has not diminished at any point in my life and I see no indication that it intends to. There are still problems I have not solved. Still frameworks I have not finished. Still questions that PP and I have opened and not yet closed. The curiosity is the engine, and the engine is still running.

Still building: yes. The Raspberry Pis on the desk are evidence. The CLI tools in various states of completion are evidence. This book is evidence. The next thing — which I can feel forming in the background, not yet legible but present, the way a process shows up in memory before it fully loads — is evidence. I will always be building something. This is not compulsion. It is identity.

Still loving: yes. More clearly, I think, than ever. The love has been through enough versions now to have shed the parts that were performance and retained the parts that are structural. What remains is reliable. What remains is the real thing, tested across decades and across losses and across all the ordinary days that are, when you look at them with sufficient attention, not ordinary at all.

Still running.

The view from within is the only one that matters.

About the Author

Marc is a technology professional, systems architect, and creator of the AxisOS philosophy. With over three decades of experience across enterprise technology, data center operations, programming, data management, and software development, he has spent a career turning complexity into clarity. He has supervised three data centers, built enterprise-grade systems from the ground up, and developed a philosophical framework for understanding human life through the lens of systems architecture.

He lives on the Oregon coast with his wife Kum-nan, building things that matter — in code, in prose, and in the lives of the people he loves. He works out of Cedar Mill and Rockaway Beach, Oregon, frequently in the small hours, frequently with a Raspberry Pi nearby.

The View From Within is his first book.

About AxisOS

AxisOS is a personal operating system philosophy built on a simple and radical premise: a human life can be understood, debugged, and optimized with the same care and discipline as any great technical system. AxisOS is not software. It is a way of seeing.

Developed by Marc across thirty years of enterprise technology, late-night build sessions, and hard-won personal experience, AxisOS offers frameworks for understanding values as kernel processes, relationships as infrastructure, losses as security events, and resilience as defensive orchestration. It approaches the examined life not as a philosophical luxury but as a practical necessity — the difference between a system that runs by accident and one that runs by design.

AxisOS acknowledges that every human system is unique, that the architecture that serves one person may not serve another, and that the most important specification document any system can maintain is an honest one. It is a philosophy built for adults who have lived enough to understand that complexity is not an obstacle to clarity but the medium through which clarity is found.

AxisOS is, like its creator, still running.

Typeset in Constantia and Cambria, from the Microsoft ClearType Font Collection.

Composed in the early morning hours on the Oregon coast, 2026.

AxisOS Press — Oregon Coast

“The view from within is the only one that matters.”