The Post-Launch Gap: Why Small-Business Websites Fail After They Go Live, and a Subscription Model for Ongoing Care

Author: Jacob Wonder **Affiliation:** Snazzy Solutions, Raleigh, North Carolina, USA
Contact: jacob@snazzy.solutions · https://jacobwonder.com **ORCID:** 0009-0005-5698-1463

Version: 1.0 (working paper) **Date:** July 18, 2026

Abstract

Small-business websites are sold, built, and launched as if launch were the finish line. It is not. The site then enters years of unmanaged operation during which software ages, attack surface grows, performance decays, content rots, and non-technical owners or staff make changes that quietly break what worked. The failures are silent until they become expensive: a hacked site, a slow site that loses rankings, a form that has not delivered a lead in months, or an outage during a sales push. I call this interval the *post-launch gap*, and I argue it is a systematically under-owned source of small and medium-sized business (SMB) website failure, distinct from the platform choice and the initial build that get most of the attention. Drawing on the maintenance and issue record of a Raleigh, NC web agency, spanning 107 websites across five hosting environments and the client rosters of five agencies it serves as a white-label maintenance partner, over more than five years, this paper (1) offers a taxonomy of the recurring post-launch failure modes and why each stays invisible until it is costly, (2) sets out the economics of neglect versus proactive care, and (3) proposes a subscription-based ongoing-care model, expressed as an evaluable service framework that owners can use to judge any provider and that agencies can use to structure a recurring-revenue maintenance line. Even the fully managed platforms that remove the technical half of this burden, such as Wix and Squarespace, leave the editorial half, namely keeping content accurate and fresh, entirely with the owner. The contribution is to reframe the SMB website as an operated system with an ongoing cost of ownership, rather than a project that ends at launch.

Keywords: small business, website maintenance, managed hosting, web operations, technical debt, recurring revenue, subscription services, managed services, white-label, total cost of ownership

JEL codes: M15 (IT Management), L86 (Information and Internet Services), L84 (Business and Professional Services), M21 (Business Economics)

---

1. Introduction

The prevailing commercial model for SMB websites treats the build as the product and the launch as the finish line. An owner pays for a site, it goes live, and the transaction is considered complete. In practice, launch is the beginning of the part that determines whether the site keeps working: years of operation during which nothing about the site is static. The software it runs on ships updates weekly. The plugins it depends on age, break, and occasionally get abandoned. Bots find and hammer whatever endpoints are exposed. Images pile up unoptimized. Staff who were never trained on the system edit pages and remove the things that made them convert. Search engines re-crawl and re-rank. None of this announces itself.

I call this interval the **post-launch gap**: the space between "the site is live" and "the site is still doing its job," which no one is contractually responsible for unless someone chose to be. My argument is that this gap, not the platform choice or the initial design, is a systematically under-owned source of SMB website failure. The failure is structurally invisible, so owners underinvest in preventing it until a visible, expensive event forces the issue.

This is not a hypothetical. In routine audits of small-business sites that were "up and running," the same findings recur: a homepage missing its H1; dozens of unoptimized images, including single photos over five megabytes; publicly exposed archive endpoints absorbing hundreds of bot requests on a server barely able to serve real visitors; and a core page-builder a full major version behind, both a security and a compatibility risk. Every one of these is a post-launch decay, not a build defect. Every one was silent to the owner.

The intended readers are two. For the **SMB owner**, this paper is an argument for treating a website as an operated asset with a running cost, and a framework for evaluating what "maintenance" should actually include before paying for it. For the **agency or practitioner**, it is a case for structuring ongoing care as a recurring-revenue service line, including as a white-label offering to other agencies that build but do not want to operate.

Related work and intellectual lineage

The reframing this paper proposes is an application of established ideas, not an invention. Treating an asset's true cost as its lifetime cost of ownership rather than its purchase price is the core of the total-cost-of-ownership literature in purchasing and supply management (Ellram, 1995); the argument here is simply that a website, sold as a one-time build, has a total cost of ownership its buyers systematically ignore. The finding that maintenance, not construction, dominates a software system's lifetime cost is long established: Lientz and Swanson's (1980) survey of 487 organizations documented maintenance as the majority of software effort and named the preventive-maintenance category this paper's care model occupies. The mechanism by which that cost accrues, the quiet accumulation of unaddressed maintenance into compounding future expense, is the software-engineering concept of technical debt, first named as a metaphor by Cunningham (1992), developed into theory and practice by Kruchten, Nord, and Ozkaya (2012), and more recently quantified in aggregate at roughly $1.5 trillion of accumulated debt in U.S. software (Krasner, 2022). The failure taxonomy in Section 3 can be read as a catalogue of the forms technical debt takes on a small-business website.

The care model in Sections 4 and 5 likewise sits within an existing tradition. The shift from selling a product to selling an ongoing service or outcome is the subject of the servitization literature introduced by Vandermerwe and Rada (1988), and the specific move to recurring, subscription-priced revenue is treated in the literature on subscription and recurring-revenue business models (Tzuo and Weisert, 2018). The contribution here is not any of these concepts individually but their application to a case the SMB-web practitioner literature has, as far as I am aware, left largely unexamined: the small-business website as an operated asset with a lifetime cost of ownership, and the maintenance market that its post-launch decay produces.

2. Scope and Method

This is a practitioner paper. Its evidence base is the maintenance and issue record of one web agency, Snazzy Solutions (founded 2017), but that record is deliberately broader than a single firm's own clients. The agency has built more than 260 websites over its history, and its current maintenance and issue data cover **107 distinct websites**, observed across **five hosting environments** (xCloud, WP Engine, SiteGround, Cloudways, and a custom server) over **more than five years** of continuous operation. Those 107 are not just its own direct clients. The agency is the **white-label maintenance and hosting provider for five other agencies**, and for some of them it manages the *entire* client roster (for example, one media agency's whole book, and all

thirty sites of one marketing agency), so its vantage point extends across multiple agencies' portfolios rather than one. This cross-agency, multi-host span is what lets the failure modes below be observed as recurring patterns rather than the quirks of a single setup: a failure that shows up on WordPress sites hosted five different ways, sold through five different agencies, is not an artifact of one host or one shop.

Of the 107 sites, roughly 75 are on a recurring plan and about 38 on full proactive maintenance rather than hosting alone; the proactive-care portfolio is maintained by Kelly Eggebeen, the agency's website quality coordinator. The agency runs its own care on infrastructure it controls, with a content-delivery and bot-mitigation layer in front of every site, and it applies updates to each site by hand rather than relying on blind fleet-wide auto-updates; the sites it observes, however, span the five hosts named above rather than one.

The agency also keeps an internal log of problems encountered during maintenance and audit work, a few hundred entries at the time of writing. It is a work record, not a controlled sample, so it cannot support population-level rates; but its shape is informative. The largest technical categories are plugin, caching, page-builder, and theme faults, followed by a tail of server, migration, content, and third-party integration failures, and the failure modes below are drawn from those recurring categories rather than assembled a priori.

Three caveats, stated plainly:

1. **Selection is not random.** The sample skews to the businesses one Raleigh agency and its agency partners serve: local service businesses, retail, food and specialty goods, small professional practices, and small manufacturers.
2. **The observer is a participant.** I sell the maintenance subscription this paper describes. The framework in Section 6 is therefore written to be *evaluable by the reader against any provider*, not as a pitch. Its test is whether it helps an owner judge a care plan, including one that is not mine.
3. **The web moves.** Specific software versions, prices, and threat patterns are snapshots that will date quickly. The *failure taxonomy* and the *care framework* are intended to outlast any particular snapshot.

The method is inductive: the failure modes in Section 3 are drawn from what actually recurs in maintenance and audit work, not assembled from an a-priori list.

3. A Taxonomy of Post-Launch Failure Modes

Each failure mode is described by what it is, why it stays invisible to the owner, and what it costs when it finally surfaces. The examples lean on WordPress, the dominant substrate for small-business sites (it powers roughly 42% of all websites, and the majority of those running a recognized content-management system; W3Techs, 2026), but the failure modes generalize to any self-managed site.

F1: Dependency and platform aging (security and update debt)

A self-hosted site is an assembly of independently versioned parts: a content-management system (CMS) core, a theme, and often dozens of plugins, each on its own release cadence and many shipping security patches. Keeping them current is not a single switch. Core releases include major version jumps that can break a theme or a plugin built against the old version. Free plugins update from the platform's own repository, but premium (paid) plugins update from the *vendor's* own servers and require an active license key to do so, so a lapsed license, a changed update endpoint, or a silent authentication failure can leave a paid component stuck on an old, vulnerable version with no visible error. And the updates themselves are a failure mode of their own: applying one can break a live site as surely as skipping one can expose it, which is why blind auto-updating is not a safe answer either. In one case in the author's records, an automatic plugin update took a neglected site completely offline. WordPress's own documentation advises taking a backup before enabling auto-updates, in case an update goes wrong (WordPress.org, 2023). *Invisible because* a site left behind keeps rendering normally right up until it is exploited, and a license that quietly stopped updating looks identical to one that is current. Industry threat reporting consistently attributes a large share of website compromises to outdated, vulnerable components of exactly this kind (Sucuri, 2024), and the scale is WordPress-specific and large: security researchers catalogued nearly 8,000 new vulnerabilities in the WordPress ecosystem in 2024, the overwhelming majority in third-party plugins, and roughly a third still unpatched at public disclosure (Patchstack, 2025; Wordfence, 2025). *Cost when it surfaces:* malware cleanup, blocklisting, lost trust, and sometimes a full rebuild. In the author's experience, this debt is the rule rather than the exception: sites the agency has taken over from prior providers have frequently had no updates applied at all, and a neglected site brought current for the first time can take hours of careful, staged work to close the accumulated gap without breaking something in the process.

F2: Performance decay

Media accumulates unoptimized (a single hero photo at multiple megabytes is common), plugins stack, and caching drifts out of tune. *Invisible because* the owner's own browser has everything cached, so the site feels fast to them; new visitors on mobile do not. *Cost:* slower pages are associated with lower conversion and weaker search performance, the latter because Google's ranking systems incorporate Core Web Vitals as a page-experience signal (Google Search Central); it is a quiet, compounding revenue leak rather than a single event.

F3: Exposed attack surface and bot load

Endpoints that serve no visitor purpose (author archives, unused feeds, open enumeration points) get discovered and hammered by bots, consuming server memory and degrading performance for real users, especially on the small hosting plans SMB sites are usually placed on. Automated traffic now makes up roughly half of all web requests, a large share of it hostile (Imperva, 2024), so an exposed endpoint is found and hit quickly rather than eventually. *Invisible because* it shows up as "the site feels slow sometimes," not as an obvious attack. *Cost:* degraded performance, resource exhaustion, and a wider breach surface.

F4: Content decay (link rot and staleness)

Two distinct things happen to content over time. It *breaks*: links die, embedded forms silently stop delivering, sitemaps go stale, pages get orphaned. The costliest instance of the breaking kind is the lead-capture form that still renders and still submits, but whose email notifications have quietly stopped arriving, whether from a broken outbound-mail (SMTP) configuration, a mail password re-encrypted by a migration, or a mail service that was never set up, so that every inquiry the form collects evaporates unseen while the form itself looks perfectly healthy. In the author's issue record, this recurs often and is among the last failures to be noticed, because the business only learns of it when it wonders why the leads stopped. In one case a single retailer lost its website leads three separate times in about seven months, each to a different silent cause (a form-breaking security misconfiguration, a mail-delivery change that killed notifications, and a malware redirect), and each was caught only when the owner asked why submissions had gone quiet. And content also *ages*: hours, pricing, staff, services, and offers drift out of date, and a site that stops publishing anything new reads as neglected to both visitors and search engines. *Invisible because* nothing errors loudly; a form that stopped emailing leads looks identical to one that works, and a page with last year's prices looks fine until a customer acts on it. *Cost:* leads and inquiries lost with no alert, plus the slow credibility

and search penalty of a site that has visibly stopped being tended. Freshness is not a break to be fixed once; it is a standing editorial task with no technical fix, which is what makes it the failure mode no platform can solve for the owner.

F5: Owner- and staff-induced breakage

Non-technical staff edit the site with no training, as one client described their own situation, with "people touching it" who "don't know how to use it," and they remove or misconfigure the elements that made it work, up to and including structural SEO elements like the homepage H1. *Invisible because* the person who made the change did not know it mattered. *Cost:* regressions in conversion and search that no one connects back to the edit that caused them.

F6: SEO and indexation decay

Rankings are not static. Without ongoing attention, technical SEO drifts, indexation issues creep in, and competitors who *are* maintaining their sites pull ahead. *Invisible because* traffic erodes gradually. *Cost:* a slow slide down the results that is far more expensive to reverse than to prevent.

F7: Compliance and policy drift

Privacy, cookie, and accessibility expectations change; an unmaintained site falls out of step. *Invisible because* nothing breaks functionally. *Cost:* legal and reputational exposure that arrives all at once.

F8: External and third-party dependency failure

Modern sites lean on services they do not host: embedded maps, web fonts, payment gateways, booking and scheduling widgets, analytics, CDNs, DNS, and assorted third-party scripts. Many are gated behind an API key tied to a separate account and its billing. When one of those *external* conditions changes, a feature breaks on a site whose own code never moved. The cause sits off the property entirely, in another vendor's console, which is exactly what makes it hard to trace: the site's own files, server, and CMS all look healthy, so every normal check passes while a visible feature is dark. *Invisible because* nothing on the site itself is wrong, so ordinary uptime and integrity monitoring report no fault. *Cost:* a broken feature (a map that will not load, a payment button that fails, a booking form that errors) that keeps losing the business customers until someone notices, and because the trigger is external it is the failure most easily mistaken for "the site is fine." The failure can also be total rather than partial. In the author's issue record, a neglected sod producer's site went fully dark more than once, from different causes;

one of them was an absent external credential: a plugin tried to initialize its payment integration at load time with a missing API key, and that single missing credential returned a server error on every page of the site. And the recurring, multi-party version is its own distinct problem: sites that break repeatedly across hosting migrations and third-party integrations handled by different hands, with no single party owning the whole surface.

The unifying property shows up in the "invisible because" line of every entry. Much post-launch failure is **asymptomatic until it is acute**, latent for a long time and then suddenly costly (F1, F3, F4, and F8 fit this shape most cleanly). Others instead degrade gradually (F2, F6) or bite at the moment of a change (F5). What all of them share is that none announces itself to the owner, and that absence of a signal is what drives the economics in Section 4.

Two kinds of upkeep, and why the platform question does not settle it. The failure modes sort, if imperfectly, into two groups. F1, F2, F3, and F7 are *technical and operational* upkeep: software, infrastructure, and security that a platform could in principle manage on the owner's behalf. F4, F5, and F6 are *editorial and human*: content that must be kept accurate and fresh, people who must not break things, and search relevance that must be tended. F8 straddles the two: it is technical in nature, but because the dependency runs through the owner's own external account, no platform fully absorbs it. This is why "just use a managed platform" is only a half-answer, a point developed in Section 7. Platforms like Wix and Squarespace can absorb most of the technical group, but nothing absorbs the editorial one, and the owner's own third-party keys (F8) remain theirs no matter how managed the underlying platform is.

4. The Economics of Neglect

Two models exist for handling the post-launch gap. Most SMBs default to the first without ever choosing it.

The reactive model (fix-when-broken). No one owns the site's operation. Problems are addressed only once they become visible, which by definition is after they have already cost something. Because the failures in Section 3 are asymptomatic until acute, the reactive model systematically intervenes at the most expensive possible moment: after the hack, after the ranking loss, after the leads have stopped. The apparent saving (paying nothing between emergencies) is often an illusion, because the emergencies tend to be more expensive than the prevention would have been, and some of them (a rebuild

after a compromise, rankings lost to a competitor) are only partly recoverable at any price.

A single case shows the magnitude this can reach. A small business ran several websites under another agency's management and was paying for their upkeep, but the maintenance was not actually being performed. All of the sites had been quietly compromised, one of them for years, and the infection intermittently redirected visitors to spam pages and left injected content of the kind Google penalizes. Because no one had registered the sites in Google Search Console, the channel that surfaces exactly these security warnings, the compromise ran undetected for more than a year while the sites' search rankings collapsed. The agency that eventually cleaned it up estimated the lost income at more than $300,000. That figure is an estimate, not an audited number, and it should be read as one; but the shape of the loss is the point. A site that looked fine to its owner, an owner who believed they were covered, and a large cost that stayed invisible until long after it began: this is the reactive model's failure mode at full size.

The proactive model (ongoing care). Someone is continuously responsible: updates are applied and tested, backups are held, monitoring watches uptime and integrity, performance is kept in tune, and the attack surface is actively reduced. How this is delivered varies by provider. In this agency's implementation, for example, it is met with over-provisioned managed hosting and a content-delivery and bot-mitigation layer on every site (a CDN with a web-application firewall materially reduces a CMS's exposure to common exploits and automated attacks; Cloudflare, n.d.), so the common failure modes have less room to bite; another provider might meet the same responsibilities on different infrastructure. Whatever the delivery, the cost to the owner is a modest, predictable recurring fee rather than an unpredictable emergency bill. In this agency's portfolio the fee falls into two tiers that the data bears out: basic managed hosting near $15 per month, and proactive maintenance, which layers active updating, monitoring, hardening, and upkeep on top of hosting, near $95 to $100 per month. That maintenance figure sits within the range vendor cost guides report for professional WordPress upkeep, roughly $50 to $200 per month (Elementor, n.d.). Across the roughly 75 sites under recurring care these appear as two distinct clusters, one near the hosting price and one near the maintenance price, with little in between, which reflects two different products: keeping a site *online* versus keeping it *well*.

The reason owners underinvest is not irrationality; it is that the value of prevention is unobservable. You cannot see the hack that did not happen or the lead that was not lost. The proactive model's real product is therefore *risk reduction and continuity*, which is exactly the kind of value that is hard to sell and easy to skip, until the first acute failure

makes it concrete. A central purpose of the framework in Section 6 is to make that invisible value legible in advance.

A single concrete case shows why the value of the proactive model is so hard to see. Lane DDS, a multi-location dental practice the agency maintains, had the store-locator map on its site stop rendering. Nothing on the website had changed. The map was a Google Maps embed whose API key was tied to a Google Cloud billing account, and that billing had lapsed and needed updating, so Google simply stopped serving the map. Every on-site check would have passed: the files were intact, the server was up, the pages loaded. The failure lived entirely off the property, in a separate vendor's billing console, and it was resolved by updating the billing card so the key came back. The point is not the fix, which was small, but the shape of the incident: a live feature that patients use to find an office went dark for a reason invisible to any normal look at the site, and closing it required someone who was actually watching the site behave in the real world, not just watching the server stay up. On the reactive model, this surfaces when a patient complains. Under active care, the patient never sees it. That gap, the outage that did not become visible, is the proactive model's real product, and it is invisible by design.

The tiers also expose a choice owners are often making without realizing it. Most sites the agency has built are on no recurring plan at all; among those that are, a substantial share buy only the hosting tier, which keeps the site *online* but leaves every failure mode in Section 3 uncovered. The distance between hosting and full maintenance is, in effect, the post-launch gap with a price on it.

5. Why the Economics Produce a Subscription Market

This section makes a descriptive claim about market structure, not a recommendation to any individual owner: whatever a given owner decides, the structure of the problem tends to produce a subscription *market*, whoever the buyer and seller turn out to be. (The paper as a whole is not neutral about whether the gap should be closed; it is neutral here only about the specific question of why the market takes this shape.)

Three properties of post-launch care push it toward a recurring contract rather than a one-off transaction. First, **the work is continuous, not discrete.** Updates arrive weekly, threats are constant, and content ages every day, so there is no single moment at which the job is "done." Continuous work maps naturally to a continuous fee. Second, **the value delivered is continuity itself.** What the buyer receives is not an artifact but a state of ongoing safety and function, which only a standing arrangement can provide; a one-time cleanup restores the state briefly and then lets it decay again. Third,

the risk is better held in a pool than by an individual. Any single site's odds of a serious failure in a given month are low but non-trivial and unpredictable; a provider maintaining many sites converts that lumpy tail risk into a smooth, priceable cost, which is the same logic that underlies insurance and other managed-service markets.

These properties also explain the market's *structure*, including the white-label layer. Because operating sites is a continuous, poolable function distinct from the discrete work of building them, it can be separated from the build and provided wholesale: agencies that build but do not want to run an operations practice can resell a specialist's ongoing care under their own brand, with a margin between the wholesale and retail price (care bought wholesale near $100 per month is commonly resold to the end client around $250). The result is a layered market in which the party that builds a site, the party that fronts the client relationship, and the party that actually operates the site need not be the same, an arrangement that is common in practice and, in the author's experience, under-examined in writing about SMB web work.

Finally, the subscription framing has a second-order effect that matters for the argument as a whole: a recurring fee makes the otherwise-invisible operational cost of a website *explicit and budgetable*. Naming a monthly number for upkeep is often the first thing that gets the post-launch gap taken seriously at all, independent of who is paid to close it, or whether an owner chooses to close it themselves.

6. A Care Framework: What "Maintenance" Should Actually Cover

"Maintenance" is sold loosely and means very different things. This framework lets an owner or a reselling agency evaluate any care plan against the failure modes it is supposed to prevent. For each capability: does the plan include it, how is it verified, and which failure mode (F1–F8) does it address?

Capability	Addresses	How to verify the plan actually does it
Updates applied and tested	F1, F5	Core, theme, and plugins updated on a schedule *and* checked afterward, not auto-updated blindly
Hosting with real headroom	F2, F3	Resources sized above the site's needs, not the cheapest shared plan it barely fits
Content delivery + bot mitigation	F2, F3	A CDN and bot-filtering layer present by default, not sold as an add-on
Backups + tested restore	F1	Backups exist *and* restores are actually verified
Uptime + integrity monitoring	F1, F3, F4	Someone/something is watching, with alerting
Performance upkeep	F2	Media optimization and cache tuning are ongoing, not one-time
Broken-link / form-delivery checks	F4	Forms are periodically confirmed to still deliver leads
Security hardening / surface reduction	F1, F3	Unused endpoints closed; exposure actively reduced
SEO/indexation health checks	F6	Technical SEO watched, not assumed static
Compliance review	F7	Privacy/cookie/accessibility kept current
Change control / owner guardrails	F5	Guidance or guardrails so staff edits do not silently break things
Third-party / integration checks	F8	Embedded maps, payment, booking, and API-keyed services confirmed live, with the external accounts and billing they depend on tracked

The framework's claim is not that every plan must include every row, but that a plan should be able to *say* which rows it covers and how, and that an owner should map the gaps to the failure modes they are choosing to leave uncovered. A one-time cleanup, by contrast, addresses none of these durably: a fragile, bloated site is not made permanently well by a single pass; it needs ongoing care.

7. Boundary Conditions

- **The very smallest, simplest sites** (a static brochure page, rarely edited, on a managed platform that absorbs updates) genuinely need less. The framework's weight scales with a site's complexity, edit frequency, and business-criticality.
- **Fully managed SaaS platforms (e.g. Wix, Squarespace)** solve most of the *technical* column. Because the vendor owns the software, hosting, security, and updates, F1 (dependency aging), F3 (attack surface), and F7 (compliance) largely disappear, and F2 (performance) is substantially handled. This is a real and honest advantage: an owner on one of these platforms is not exposed to the silent security-and-update decay that unmanaged self-hosted sites suffer, which is a legitimate reason to choose them. But it solves only half the problem. The *editorial and human* column is untouched: content still has to be kept accurate and fresh (F4), staff can still break things (F5), and search relevance still has to be tended (F6). A Squarespace site that is never updated is safe from hacks but still goes stale, still shows last year's information, and still slides in search. No platform makes a website "set and forget"; managed platforms move the remaining work from the server to the words; they do not remove it.
- **Automated update-and-verify services** (offered by managed WordPress hosts such as WP Engine, and by some maintenance tools) genuinely help with F1. They apply updates on a schedule and run before-and-after checks to flag when an update breaks something: WP Engine, for example, runs automated smoke tests before and after each core update and rolls the update back automatically if the post-update check fails (WP Engine, n.d.), and other tools add visual before-and-after snapshots. This catches many regressions automatically and saves a large amount of routine time. But automation reduces the burden; it does not remove it. In the author's experience across large agency fleets, most sites will at some point hit an update that breaks something the automated check cannot fully resolve, and a person still has to diagnose and fix it. The service changes the job from applying every update by hand to supervising the exceptions. The human does not disappear. This is the same pattern as the managed-platform case above: the tooling absorbs the routine; the judgment stays with a person.
- **Owner capability matters.** A technically capable owner can carry some of this themselves; most SMB owners in the observed sample cannot and should not try.

8. Limitations

The evidence base is broad for a practitioner account: 107 sites across five hosts and the rosters of five agencies, over more than five years, which is what lets the failure modes read as recurring patterns rather than one shop's quirks. But breadth is not representativeness. It remains a convenience sample, not a random one: it reflects the sites that flow through one maintenance provider and its agency partners, and it skews toward sites that arrived already troubled (selection and survivorship bias). Participant-observer interest applies, since the author sells the service analyzed here, as does a fast-moving domain. The paper's claim is therefore structural and procedural (that the post-launch gap is a systematically under-owned source of SMB website failure, and that this framework helps reason about covering it), not an empirical measurement of failure rates. Where it states patterns, they are observed in practice and drawn from an internal issue log, not sampled to a statistical standard. Turning those observations into measured failure rates, on a defined denominator of sites over a defined period, is the obvious next step and is not attempted here.

9. Conclusion

The SMB web industry sells websites as projects and leaves them to run as unowned systems. The result is the post-launch gap: a long interval of silent decay that, left unowned, can lead to expensive and sometimes only partly reversible failure. The response is neither a better build nor a smarter platform choice; it is treating the site as an operated asset with a running cost, which the market tends to deliver as ongoing care priced as a subscription, sometimes white-label through the agencies that build but do not operate. This paper's contribution is to name the gap, to explain why it stays invisible until it is costly, and to give owners and agencies a framework for deciding, in advance, how much of it to cover. Measuring how often, and how expensively, the gap actually bites remains open work.

References

Cloudflare. (n.d.). *Improving web security for content management systems like WordPress.* Cloudflare Docs. Retrieved 2026, from https://developers.cloudflare.com/support/third-party-software/content-management-system-cms/improving-web-security-for-content-management-systems-like-wordpress/

Cunningham, W. (1992). The WyCash Portfolio Management System. *Addendum to the Proceedings of OOPSLA '92*. Association for Computing Machinery. http://c2.com/doc/oopsla92.html

Elementor. (n.d.). *WordPress website cost guide*. Elementor Blog. Retrieved 2026, from https://elementor.com/blog/wordpress-website-cost-guide/ (Vendor cost guide; cited for typical market pricing, not as independent research.)

Ellram, L. M. (1995). Total cost of ownership: An analysis approach for purchasing. *International Journal of Physical Distribution & Logistics Management*, 25(8), 4–23. https://doi.org/10.1108/09600039510099928

Google Search Central. *Understanding Core Web Vitals and Google search results*. Google for Developers. https://developers.google.com/search/docs/appearance/core-web-vitals

Imperva. (2024). *2024 Bad Bot Report*. Imperva (Thales). https://www.imperva.com/resources/resource-library/reports/2024-bad-bot-report/

Krasner, H. (2022). *The Cost of Poor Software Quality in the U.S.: A 2022 Report*. Consortium for Information & Software Quality (CISQ). https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software*, 29(6), 18–21. https://doi.org/10.1109/MS.2012.167

Lientz, B. P., & Swanson, E. B. (1980). *Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations*. Addison-Wesley.

Patchstack. (2025). *State of WordPress Security in 2025*. Patchstack, in partnership with Sucuri. https://patchstack.com/whitepaper/state-of-wordpress-security-in-2025/

Sucuri. (2024). *2023 Hacked Website & Malware Threat Report*. Sucuri / GoDaddy. https://sucuri.net/reports/2023-hacked-website-report/

Tzuo, T., & Weisert, G. (2018). *Subscribed: Why the Subscription Model Will Be Your Company's Future, and What to Do About It*. Portfolio / Penguin.

Vandermerwe, S., & Rada, J. (1988). Servitization of business: Adding value by adding services. *European Management Journal*, 6(4), 314–324. https://doi.org/10.1016/0263-2373(88)90033-3

W3Techs. (2026). *Usage statistics and market share of WordPress*. Q-Success. https://w3techs.com/technologies/details/cm-wordpress

Wordfence. (2025). *2024 Annual WordPress Security Report*. Wordfence (Defiant, Inc.). https://www.wordfence.com/blog/2025/04/2024-annual-wordpress-security-report-by-wordfence/

WordPress.org. (2023). *Plugin and themes auto-updates*. WordPress Documentation. https://wordpress.org/documentation/article/plugins-themes-auto-updates/

WP Engine. (n.d.). *WordPress core automatic updates and deferrals*. WP Engine Support. Retrieved 2026, from https://wpengine.com/support/wordpress-updates/

Author Note

Jacob Wonder is the founder of Snazzy Solutions (2017), a Raleigh, North Carolina web agency that builds, hosts, and maintains websites for small businesses and as a white-label partner to other agencies. He built his first website in 2008 and has run Snazzy Solutions since 2017.

Conflict of interest. The author owns and operates a business that sells the maintenance service this paper analyzes. The framework in Section 6 is written to be applied against any provider, and the paper's claims are structural rather than promotional, but readers should weigh the author's commercial interest accordingly. This is a practitioner working paper and has not undergone peer review.