

Technical SEO in the Age of Agentic AI

From a page to a received inquiry

A practical chapter and evidence worksheet

Michael C. Hurley

EDITED SAMPLE / SEPTEMBER 2026

Page

— Facts

— Action

— Delivery

— Evidence

For website owners, agency teams, and developers

A method you can apply to one important page today.

A website can look finished and still fail at its most useful job. A visitor finds the right service, understands the offer, and sends an inquiry. The page says thank you. The business never receives it.

That failure belongs in a technical review just as much as a broken link or a missing page. Search visibility brings someone to the door. The rest of the system has to work when they arrive.

This chapter gives you a small, complete unit of work: examine one important page and follow one action through to its destination. You will collect evidence, choose a repair, and repeat the same checks afterward. The result is a record of what works and what remains uncertain.

The method also helps when a person delegates part of the visit to a browsing agent. The agent may inspect a rendered page, its document structure, or its accessibility tree. Clear facts, recognizable controls, and understandable states make that visit easier to test. This is a practical engineering argument, not a promise that any particular model will recommend your business.

1. Define the path before opening a tool

Choose one page that matters to an existing business. A service page with a request form is a useful starting point. So is a product page with a purchase action or a restaurant page with a reservation link. Pick a path that has an owner who can confirm what happens at the other end.

Write down the visitor's question and intended action in ordinary language. For example: "A homeowner wants to know whether this company repairs well pumps in their town and how to request a visit." That sentence tells you which facts must be visible. It also prevents the review from expanding into every possible improvement to the website.

Then record five things:

1. The starting URL and the service or product it describes.
2. The facts someone needs before taking the action.
3. The primary action, including what it commits the visitor to.
4. The receiving system and the person responsible for responding.
5. The evidence that would prove the action completed.

A confirmation message is evidence of a visible state. A record arriving in the intended inbox or CRM is evidence of delivery. An analytics event is evidence of measurement. Treat them as separate checks, because each can succeed while another fails.

Agree on the test scope with the person responsible for the site. Reading public pages is different from sending inquiries, reserving appointments, or placing orders. For an end-to-end form test, identify it as a test, use an agreed destination or test account, and arrange cleanup. Keep real customer information out of screenshots and sample reports.

Your first deliverable: a single sentence describing the path, followed by its start, action, destination, owner, and proof of completion.

2. Check what is delivered, then what is rendered

Start with the response to the page request. Record the time, final URL, status code, and any redirects. An unexpected redirect to a generic homepage may be more important than a minor formatting issue. A success status alone is not enough: the response could contain an error message, a login screen, or a mostly empty application shell.

Read the initial HTML and then inspect the page in a browser. Ask where the essential facts appear. Can you find the business name, service, service area, and contact route in the response? If a fact appears only after JavaScript runs, record that difference. It is an observation to investigate, not automatic proof of an indexing failure.

Google documents a process of crawling, rendering, and indexing for JavaScript pages. It can execute JavaScript, while other consumers may have different capabilities. Server-side or static delivery of essential content can reduce dependencies, but the right rendering choice depends on the page and its users. [Google's JavaScript SEO basics](#).

Inspect the browser's document structure as well as the screenshot. Text can look like a heading without being marked as one. A contact action can look like a button while being implemented as an unlabeled generic element. These choices can affect how someone using assistive technology, a testing tool, or an agent understands the page.

The web.dev guidance on agent-friendly websites describes screenshots, HTML, and the accessibility tree as complementary representations. It recommends semantic controls, associated form labels, and stable interfaces. Those are useful things to inspect directly rather than infer from the page's appearance. [Build agent-friendly websites](#).

For this review, collect three observations:

- **Response:** the URL, status, and essential facts returned by the server.
- **Browser:** the facts and actions available after the page renders.
- **Interaction:** the names, roles, labels, and states of the controls needed to continue.

Do not rename a browser test “a simulation of Google” or “a Perplexity audit” because you changed its user-agent string. A request header does not reproduce another company's retrieval system, rendering decisions, ranking, or agent behavior. Give the test the name of the actual tool and conditions you used.

3. Make the facts agree

The page should explain who provides the service, where it is available, what the next step involves, and any material limits on the offer. The same facts should remain consistent in the footer, contact page, business profile, and structured data where they appear.

Suppose a page says appointments are available only in one county, while its markup lists five states. Adding more location names does not make the information more useful. It creates a contradiction. Resolve the source of truth with the business owner and correct each affected representation.

Structured data can describe the page's entities and relationships explicitly. It should reflect the content people can see and should use an appropriate type. Passing a syntax check does not prove the information is true or that a search feature will appear. Google's structured data guidelines separate technical validity from content quality and eligibility. [General structured data guidelines](#).

Use this compact fact check before adding markup:

- Does the business name match the current public identity?
- Are the service, location, hours, and offer terms accurate?
- Does the visitor see the facts that the markup describes?
- Do relevant links lead to the intended pages?
- Can a named person update these facts when the business changes?

Avoid inserting placeholder people, fabricated ratings, invented telephone numbers, or unverified affiliations into a live page. A syntactically valid example remains an example until its facts are replaced and checked.

Platform features also change. Google stopped showing FAQ rich results starting May 7, 2026 and removed that feature's documentation in June. A useful questions-and-answers section may still help readers, but an old screenshot of an FAQ search feature is not evidence of current eligibility. [Google Search documentation updates](#).

There is a similar distinction for [llms.txt](#). Its usefulness to a specific system should be evaluated on that system's behavior. Google's current guidance says these files do not affect its search visibility or rankings; it also says there is no special structured-data markup required for generative search. Keep optional experiments separate from the work needed to make the page useful. [Google's generative AI optimization guide](#).

4. Follow the action to the receiving system

Now take the visitor's next step. On a phone-sized viewport, can you identify the intended contact action without guessing? Does its label describe what happens? Does it stay reachable when banners, menus, or dialogs are open?

For a form, inspect the fields and their labels. Try the agreed validation cases, such as submitting an empty required field or an incorrectly formatted email address. A useful error points to the affected field, explains what needs to change, and preserves the information already entered.

For the agreed successful test, record a unique marker such as [TEST-2026-09-08-A](#). Put it in a field intended for text, and tell the receiving owner what to expect. The marker connects the visible submission to the server record and destination without copying someone else's inquiry.

Check the following stages individually:

1. **Submission:** the interface shows a truthful pending or completed state.
2. **Acceptance:** the receiving application accepted or stored the request.
3. **Routing:** the inquiry reached the intended inbox, CRM record, queue, or booking system.
4. **Measurement:** the expected event was recorded under the intended conditions.
5. **Ownership:** the responsible person can find the inquiry and knows the response process.

The exact evidence depends on the implementation. A server log entry can show an accepted request. An email provider may show delivery status. A CRM may hold the durable inquiry record. State what each item proves. “Accepted by the mail service” is not the same as “seen by the salesperson.”

Check failure behavior as well as the successful path. If the form cannot submit, does it say so clearly? If there is a fallback telephone number or email link, is it current? In a staging environment, a controlled delivery failure can reveal whether the interface gives a false success message. Do not deliberately break a production service to run that test.

Be especially careful with retries. A double click, slow connection, or refreshed confirmation page can expose duplicate inquiries or duplicate analytics events. Note whether the system has a durable request identifier and whether its behavior matches the agreed requirement. Different systems may legitimately record more than one technical event; the business report still needs a clear definition of one inquiry.

An analytics dashboard does not settle all of this. Some visitors decline tracking or use tools that block it. A recorded event can exist without delivery, and a delivered inquiry can exist without a browser analytics event. Reconcile the sources instead of treating one counter as the entire truth.

5. Use evidence to choose the repair

It is tempting to collect every warning a tool can produce. That usually leaves the owner with a long report and no clear first step. Organize findings around the selected path instead.

Give each finding four parts: the observed behavior, its effect on this path, a proposed change, and the evidence needed to verify the change. Include “unknown” when access or testing is missing. A blank result is not a pass.

The following example is fictional. It illustrates the method; it is not a report about a named client or a claim of increased sales.

Example: Ridge Pump Service

A service page returns a successful response and clearly describes well-pump repair. On a phone, the request form accepts the agreed test. The page displays a success message, but the office cannot find the inquiry. Inspection shows that the configured destination belongs to an obsolete mailbox.

The repair is specific: update the destination through the normal release process, repeat the same identified test, and confirm receipt in the current office inbox. Keep a record of the original setting in the private change history, along with the rollback procedure. The public case narrative does not need the private mailbox addresses.

After the repair, the evidence record might say: “Test A displayed confirmation but did not reach the current office inbox. Test B, after the destination change, reached that inbox and was acknowledged by the office owner.” If analytics also passed, record that separately.

That supports a claim that the tested delivery route was corrected. It does not establish how many genuine inquiries were previously lost, how much revenue the change created, or whether future visits will convert. Those questions require additional records and time.

The same reasoning applies to performance work. A laboratory test is useful for diagnosis under recorded conditions. Field measurements describe actual visits in the available dataset. Core Web Vitals use LCP, INP, and CLS; the current good thresholds are 2.5 seconds, 200 milliseconds, and 0.1 respectively, assessed at the 75th percentile. A single fast lab run is not a field pass. [Web Vitals](#).

Choose the smallest change that addresses the observed failure without creating unnecessary work elsewhere. Record what you changed, who released it, and how to reverse it. Then repeat the baseline procedure under comparable conditions. If conditions changed, explain the difference.

6. Leave a record another person can use

A useful handoff should let someone else repeat the check. It should also make clear which conclusions the evidence supports.

For each test, keep the timestamp, URL, environment, device or viewport, action, expected result, observed result, and evidence location. Use a short status vocabulary: **pass**, **fail**, **blocked**, or **not tested**. Keep sensitive screenshots and logs in the client's approved storage; share redacted copies where appropriate.

Separate three levels of reporting:

- **Technical result:** the tested page and action behaved as expected under stated conditions.
- **Observed business activity:** actual inquiries, qualified opportunities, or sales recorded during a defined period.
- **Causal conclusion:** whether the change produced an increase, supported by an appropriate comparison and the limits of that comparison.

A before-and-after screenshot can support the first level. It usually cannot establish the third. A citation in an AI answer is another observation with its own context: record the question, platform, date, answer, and source link. Do not turn one answer into a market-wide visibility percentage.

End the handoff with the next check, its owner, and the event that should trigger it. A form destination deserves another check after a staff change. A service page deserves one after a

pricing or service-area change. A release touching forms or analytics should include a replay of the selected path.

The value of this method is a shorter distance between a technical observation and a useful action. You know which page you examined, what the visitor needed, where the inquiry went, what changed, and what remains uncertain. That is a foundation a website team can maintain.

Field worksheet

Copy these prompts into a project note before the next review.

Path: A visitor starts at _____, needs to know _____, takes the action _____, and expects _____.

Scope: Environment _____. Owner _____. Test submission arrangement _____. Excluded actions _____.

Baseline: Final URL/status _____. Initial HTML facts _____. Rendered facts _____. Labels and controls _____. Destination proof _____. Measurement proof _____.

Finding: Observed behavior _____. Effect on this path _____. Proposed repair _____. Evidence needed _____.

Replay: Change reference _____. Timestamp _____. Test identifier _____. Observed result _____. Pass/fail/blocked/not tested _____.

Handoff: Receiving owner _____. Next check _____. Trigger _____. Remaining uncertainty _____.

This edited sample draws on Michael C. Hurley's technical SEO manuscript and course. The worked business example is fictional. Platform documentation was checked September 8, 2026; consult the linked primary sources when applying instructions after that date. This chapter teaches a verification method and does not promise rankings, citations, leads, or revenue.