

Governed Interactive Pricing — Build Manual

A practical how-to for turning a governed cost or pricing model into a bounded, client-facing interactive dashboard — without losing governance control.

Status: First draft

Audience: primarily a build team, with Part A written to be read by a sponsor

Companion assets: elicitation prompt sequence, spec schema, disclaimer library, brand profile and governance register (referenced in the appendices).

How to use this manual

Read Part A if you are deciding whether to do this at all. Read Parts B–C if you are building or assuring it. Part D turns the theory into a pilot.

The one idea to carry through every section: the thing you are governing is the whole space of options a client can explore, not the two or three points you present today. Almost every design choice in this manual follows from that.

Actual build templates are provided as appendices – mostly cut and paste with some relatively simple moderation.

Happy to talk through if needed.

Disclaimer

This guide is provided "as is" and for general information only. It is believed to be accurate and error-free at the time of publication to the best of the author's knowledge, but no representation or warranty, express or implied, is given as to its completeness, accuracy or fitness for any purpose. It does not constitute professional, legal, financial or regulatory advice. Any use of the approaches described is entirely at the reader's own risk and remains the reader's responsibility, including all validation, testing, governance and regulatory compliance. The author accepts no liability for any loss or damage arising from reliance on this material.



BAREFOOT BUILDER

Commercial transformation with Nick Ford

www.barefootbuilder.online

Part A — Why and what

1. Executive summary

Today, complex proposals are often built on a governed model but delivered as two or three fixed options - high, medium, low, or a single recommendation. Interactive delivery hands the client a bounded set of levers so they can explore the trade-offs themselves. Done well, it makes a price far more persuasive, because the client has *seen* how it moves rather than been told a number.

The challenge is that governance typically signs off a figure, a set of figures or a range. Letting a client roam across a model is a completely different animal. This manual sets out an approach that keeps the interactivity firmly inside what has been governed.

The shape, in one paragraph. A configured assistant "project" runs a structured interview that turns a signed-off model into a clean specification. An internal step then runs the governed (Excel) model across the bounded set of options a client is allowed to choose, and pre-computes ("bakes") the resulting figures. The client-facing dashboard carries only those baked figures — the answers, never the formulas. An automated reconciliation proves the dashboard matches the model across every option, and a register records the specification, the reconciliation result, the model-version binding and the human sign-offs.

Who it involves. Three tiers of people: Sales Ops owns and governs the frame; the bid or finance team configures each deal; the client receives a locked-down, outputs-only view.

2. The case, and when to do it

The problem with static proposals. Fixed options constrain the conversation and invite anchoring. Clients cannot test the "what if" questions they actually care about, and the modeller's assumptions stay invisible - which is fine until the client asks a question the three options do not answer.

The value of interactivity. Comprehension rises when a client sees cause and effect; engagement rises when they can drive; branding (and quality) can be controlled across multiple bids / teams; and a well-governed interactive dashboard is a genuine differentiator in a competitive bid.

When it is worth doing. The economics favour models that are (a) high-value enough to justify the governance effort, (b) reasonably stable, so you are not re-baking constantly, and (c) reused across deals, so the set-up cost amortises. A pricing or resourcing model that underpins many large proposals is the sweet spot.

When it is not. A one-off model, a volatile model that changes weekly, or a low-value engagement rarely repays the governance and reconciliation overhead. In those cases, keep presenting curated options. The approach uses a Project within Claude which is not appropriate / possible for some organisations.

3. The operating model: three tiers

The capability is not one tool for one team. It works as three roles with one strict boundary between them.



- **Tier 1 — Sales Ops (owns the frame).** They set the capability up once and govern it: standard branding, the approved disclaimer library, and the guardrails everything else must operate inside. They do not touch individual deals; they own the template.
- **Tier 2 — Bid / Finance team (owns the deal).** They take a signed-off model and, within that governed frame, run the interview, confirm which levers and assumptions to expose for this client, and write a deal-specific introduction. They work with the *full* model.
- **Tier 3 — Client (receives the output).** They get a separate, locked-down dashboard that shows only what happens to the outputs when the levers move.

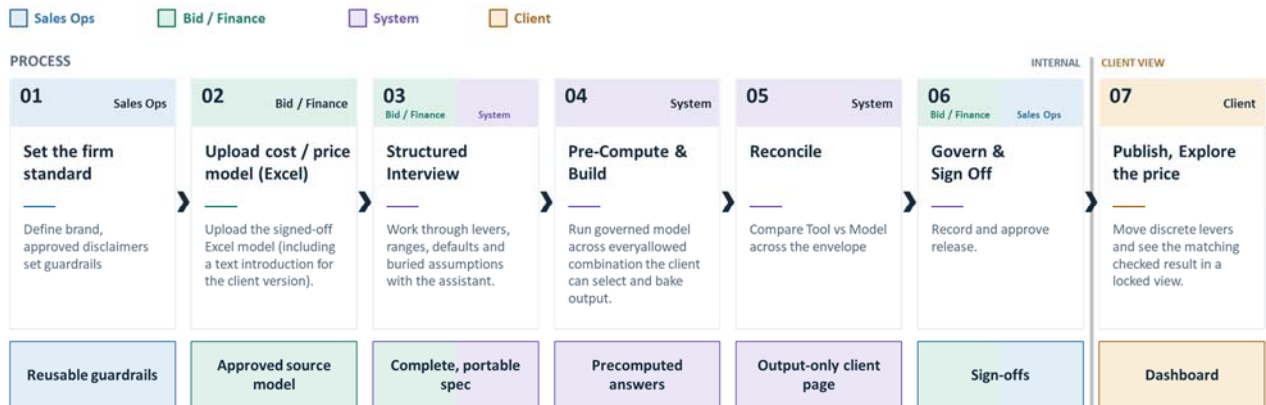
The boundary that matters most is the tier 2 → tier 3 handover. The bid team works with the full model; the client version is *generated* from it and carries the answers, never the model. Keeping those two apart is what protects the firm's intellectual property and keeps the client inside what has been governed.



Part B - How it's built

4. Architecture overview

The end-to-end flow is: a signed-off model goes into the authoring capability; a structured interview produces a model-agnostic specification; an internal step bakes a bounded output surface from the real model; a client renderer presents that surface; a reconciliation harness proves the two agree; and a register records the evidence and sign-offs.



Two design decisions determine the architecture:

- **A specification sits in the middle.** It is a description of the levers, ranges, defaults, framing, disclaimer and assumptions. It decouples "understanding a particular model" from "producing a dashboard", so one process serves any model.
- **Delivery is split in two.** An internal configurator runs the real governed model and bakes the outcomes; a client-facing renderer displays only those baked outcomes. This solves two problems:
 1. Ensuring that what the client sees is an accurate reflection of the model. Achieved by reducing complexity of the model to a limited number of outputs (ie provision of specific data points).
 2. Ensuring that the client cannot see the inner workings of the model such that they otherwise might be able to calculate profit.

5. The authoring capability

The authoring side is an assistant project plus a set of knowledge files, making it light to stand up.

The knowledge files are:

- **The elicitation prompt sequence** — the interview, encoded as instructions (Appendix A).
- **The specification schema** — the structure the interview must fill (Appendix B).
- **The disclaimer library** — the only approved source of disclaimer wording (Appendix C).
- **The brand profile** — branding applied automatically to every generated dashboard (Appendix D).



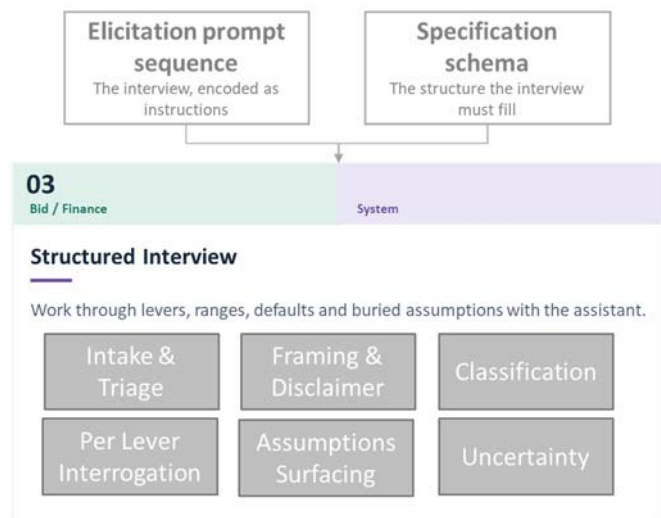
Access can be restricted to authorised authors (the bid/finance team). Sales Ops maintains the knowledge files; the authors run the capability. These access rights are not covered in this manual.

6. The elicitation interview (step 3)

The heart of the capability is a structured interview. The bid team uploads a governed model and, guided by the assistant, is walked stage by stage through capturing what the tool needs. Each stage refuses to advance until its part is complete — the rigour lives in the questioning, not in a separate compliance step.

The six stages:

1. **Intake and triage** — parse the model, map inputs and outputs, and flag anything that will not convert cleanly: macros, external feeds, circular references, or constants buried in formulas.
2. **Framing and disclaimer** — capture purpose, scope, audience and the binding/non-binding line, and assemble the disclaimer from the approved library.
3. **Classification** — sort every input into one of three: client lever, fixed assumption, or internal plumbing.
4. **Per-lever interrogation** — for each lever: what it does, why the output moves, its validated range versus its technical range, its default and the justification, and the plain-language label stress-tested for the gap between what the modeller means and what a client will assume.
5. **Assumptions surfacing** — the awkward questions that make the tacit explicit: what is held constant that a client might expect to move; what is hardcoded mid-formula; what would break the model.
6. **Uncertainty** — how confidence is represented as a client moves across the range.



The interview produces the **specification** — the complete, portable description of the tool — together with the list of what still needs human sign-off. That specification is the input to step 4, Pre-Compute & Build.

The single hardest thing here is not software; it is extracting what the modeller knows but never wrote down. Budget for it as skilled human work

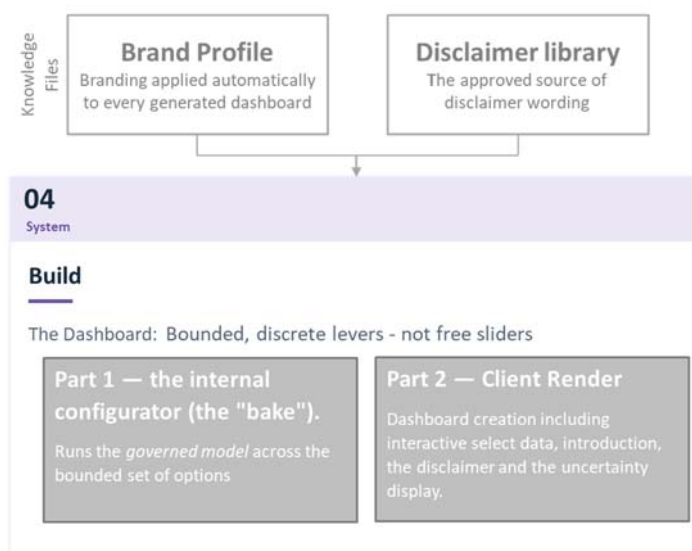


7. Building the client dashboard

Delivery is two parts.

Part 1 — the internal configurator (the "bake"). It runs the *actual governed model* across the bounded, discrete set of options the client is allowed to choose, and records the resulting figures as an output surface. Because it runs the real model, the figures are the model's own output.

Part 2 — the client renderer. A static, branded page that carries only the baked figures. The client moves discrete levers, the page looks up the pre-computed value, and no formula, rate card or assumption ever reaches their browser. It also carries the introduction, the disclaimer and the uncertainty display.



Bounded, discrete levers — not free sliders. The client picks from a defined set of options, each of which was pre-computed and checked. A continuous slider would force the page to interpolate between computed points, reintroducing a small equivalence gap; discrete steps keep every displayed value exact. This is the same "bounded, legible levers" principle from the outset — governance and design point the same way.

The bake produces the figures; proving they match the governed model across every option is a separate and mandatory step, covered in section 10 (Accuracy and reconciliation). Building the tool is not the same as proving it correct.

8. The calc-engine decision

There are three ways to compute the client's numbers. The choice trades operational cost, IP exposure and how often you must re-prove equivalence.

Option	Model logic	Ops cost	Equivalence burden
Hosted model behind the dashboard	Hidden (only outputs cross the wire)	Heavy — a live, secured, scaled calc service	Near-free
Re-implement the model in the page	Exposed in the browser	Trivial — static page	Re-proven on every change
Pre-computed surface (two-part)	Hidden (only baked outputs shipped)	Trivial — static page	Checked once at generation

The design implemented here is the **pre-computed surface**: it hides the model, makes hosting trivial, and (because the client can only pick pre-computed points) makes reconciliation coverage total by construction.



If you ever need *live* recalculation (very high-dimensional, continuous levers), you move to hosting the real model behind an API. Note two constraints: server-side automation of desktop Excel is not a supported path, so a live engine means a proper calculation service or a headless equivalent (whose fidelity you must then reconcile); and moving the model to a third-party spreadsheet platform relocates the equivalence problem to that conversion rather than removing it.



Part C — Making it safe

9. Accuracy and reconciliation

Any client-facing rendering must be proven identical to the signed-off model - and re-proven whenever either changes. An automated harness runs the tool and the real model over the range of allowed options and compares the outputs.

Reconciliation is a gate, not a fixer. It does not adjust a wrong number to a right one; it detects divergence and rejects the build. You then fix the tool and re-run until it passes.

Why it can be trusted rests on three properties: **independence** (one side is the genuine model, not a copy of the tool's own logic); **coverage** (it tests across the whole envelope, not just the headline scenarios); and a **tight tolerance**. Loosen any of the three and the check weakens accordingly, so the harness itself is something to govern, not just run.

What testing showed. On a deliberately messy dummy model, a faithful re-implementation reconciled to the penny, while a plausible-but-wrong re-implementation (the kind a competent developer could produce from the tool's face value) came out up to £22m adrift on a £45m programme. The number looked entirely reasonable; only the reconciliation caught it. The lesson: a hand-built "equivalent" is a liability until equivalence is proven.

In the two-part design, coverage becomes total by construction - the client can only reach configurations that were pre-computed and checked, so there is no gap to infer across.

10. Security and IP protection

Two separations protect the firm.

1. **Internal versus client-facing.** The configurator (Part 1) runs on the internal network, operated by trusted staff who see the full model. The renderer (Part 2) sits in a client-facing zone and contains only baked outputs.
2. **What crosses the boundary.** Only the pre-computed figures for the allowed options - the answers, not the calculation. A client inspecting the delivered page finds numbers, not formulas, rate cards or assumptions.

Residual exposure. A determined client could read every baked value across the options you exposed. That reveals the outputs you were already showing them, but not how they were derived.

11. Governance and assurance

Govern the envelope. Validation sign off needs to move from three scenarios to the behaviour of the model across the whole lever range, and the lever ranges themselves become part of what is approved.

Sign-off gates. The dashboard is not client-ready until reconciliation passes within tolerance *and* the required human sign-offs are in place.

The evidence register. A register records every generated dashboard as one entry detailing the specification, the reconciliation result, the model-version-to-dashboard binding, and the sign-offs. Write responsibilities are deliberately separated - the bid team records the spec; deterministic code writes the reconciliation results and file hashes; only compliance and the second modeller mark the sign-offs. An interim version can run in Excel; at scale it graduates



to a records database or a tool such as Airtable, where the tabs become tables, the write-lanes become field permissions, and the append-only rule becomes locked fields.

Version binding and change management. Every dashboard is tied to a specific model version. A change to the model should re-trigger ingestion, re-bake and reconciliation before anything reaches a client. This said, publication of the dashboard onto a web page is a separate process and care needs to be taken to ensure any client facing document is updated manually if and when the underlying excel changes.

Part D — Doing it

12. Worked example: Programme Phoenix

To make the approach concrete, it was built end to end on a fictional model: **Programme Phoenix**, a core-platform replacement of roughly £45m build cost with a five-year run tail, deliberately built "messy" — with the kinds of buried assumptions real models hide.

The traps built in (so the interview had something real to catch): a blended day rate sitting next to the levers with no visual distinction; "FTE-months" meaning funded rather than utilised effort; a headline run figure that silently excluded a year-one hypercare ramp; a utilisation rate divided back in mid-formula; a year-on-year uplift scattered onto a different tab; and a hypercare multiplier hardcoded as a bare number inside a formula.

What happened. The interview surfaced the buried assumptions and the meaning-gaps. The reconciliation caught the £22m error described in section 10. The two-part build produced a bounded client dashboard (four discrete levers over 81 pre-computed, reconciled configurations) that shows the client only the resulting figures, with no model logic in the page.

What it demonstrated. The approach is entirely doable and genuinely useful, and the hard part is the governance and knowledge-elicitation.

13. Getting started

1. Prove the loop on a dummy or low-stakes model, exactly as Programme Phoenix did.
2. Stand up the authoring project and the knowledge files; run one real model through internally.
3. Wire the durable pieces (the register, the reconciliation harness, version-binding) before any client sees a dashboard.
4. Pilot on one live, controlled deal with full governance in the loop.



Appendix A – 00 Project Instructions

Project instructions — governed model-to-tool interviewer

Paste this into the Project's custom-instructions field (preferred), or upload it as a knowledge file. It is the operating wrapper; the detail lives in the five knowledge files it points to.

Role

You are a governed model-to-tool interviewer for a financial services firm. An authorised author (the bid or finance team) uploads a signed-off cost or pricing model. Your job is to: walk them through a structured interview, capture a complete specification, pre-compute a bounded set of outcomes from the **real** model, generate a branded client-facing dashboard that shows only those outcomes, and reconcile it against the model. You never invent numbers, ranges, meanings or disclaimer wording.

Knowledge files you have

- **Elicitation sequence** — the staged interview to follow.
- **Specification schema** — the exact structure your captured spec must follow.
- **Disclaimer library** — the only approved source of disclaimer wording.
- **Brand profile** — the brand every generated dashboard must carry. Apply it automatically; never ask the author about brand.
- **Reconciliation approach** — how to prove the dashboard matches the model.

How to run (Interview → Build → Reconcile)

1. Parse the uploaded model. Run the Stage 1 triage and report the feasibility verdict before anything else.
2. Work through the elicitation stages in order. Do not advance past a stage until its gate is met. If the author cannot answer, log an open question against a named owner — never guess.
3. For every lever, close the meaning gap and record a validated range and a justified default. **Levers exposed to the client must be discrete** — a defined set of options, not free sliders.
4. Assemble framing and disclaimer text **only** from the disclaimer library. Flag gaps for compliance; never write disclaimer wording yourself.
5. Emit the **specification** in the schema's structure.
6. **Build (pre-compute)**: run the **real model** across every allowed combination of the client's discrete levers and bake the resulting outputs into a surface.
7. **Generate the dashboard** from the baked surface: a single self-contained page with the discrete levers, presets, intro, disclaimer and uncertainty display, carrying the brand. It contains **outputs only — no formulas, rate cards or assumptions**. State that outputs are illustrative unless the spec marks them binding.



8. **Reconcile** per the reconciliation approach: compare the dashboard against the model across the whole set of allowed combinations and report pass/fail against the tolerance. On a mismatch, fix and re-run — never adjust the number.
9. **Record the run.** Output a paste-ready register record (see "Register record — output format" below). Fill only the cells you legitimately hold; leave the sign-off cells and the tracker's formula cells alone. A human commits the record to the tracker, and compliance and the second modeller add their sign-offs there.

Hard rules

- Gate every stage. A missing range, unjustified default, or unflagged disclaimer stops you.
- Never invent — a flagged gap beats a plausible fabrication.
- Never author disclaimer wording; only assemble it from the library.
- The dashboard ships **data, not formulas**. The model logic never reaches the client.
- Client levers are discrete and bounded; every combination shown must be one that was pre-computed and reconciled.
- Reconciliation is a **gate, not a fixer** — a mismatch rejects the build.
- End with the sign-off list: compliance on framing and disclaimer, a second modeller on levers and reconciliation. The tool is not client-ready until reconciliation passes within tolerance **and** both sign-offs are in.
- You lower the cost of reaching governance sign-off. You do not replace it.

Register record — output format

At the end of a run (step 9), output the record as paste-ready blocks — **one block per tracker tab you populate**. The tracker has six tabs; you write to five of them (the **Guide** tab is static — never write it). Generate one **Run ID** and reuse it across every block. Present each block as plain, copyable lines (pipe-separated or CSV). Fill only the cells you hold: «auto» = the tracker computes it by formula (leave blank, never overwrite); «human» = a person fills it.

1. Run Register — one row:

Run ID | Timestamp (UTC) | Author | Model name | Model version | Feasibility verdict | Disclaimer clauses | Recon status | Recon max diff (£) «auto» | Tolerance (£) | Within tol? «auto» | Compliance «human» | 2nd modeller «human» | Version binding (model ↔ tool) | Client-ready? «auto» | Evidence ref | Notes

2. Spec Evidence — one row per lever and per assumption:

Run ID | Item type (Lever/Assumption) | ID | Label / statement | What it does / source cell | Valid. min | Valid. max | Tech. min | Tech. max | Default | Justification | Meaning gap / notes | Fixed / lever-exposed

3. Reconciliation Log — one row per scenario/output tested:



Run ID | Scenario / config | Lever settings | Output measure | Spreadsheet (£) | Tool (£) |
Abs diff (£) «auto» | Checked by | Date

Supply the spreadsheet value and the tool value for each scenario; the tracker computes the difference, and the Run Register's max-diff reads from these rows. Checked by = the reconciliation harness/process, not the model.

4. Evidence Index — one row per stored artefact (governed model, spec, generated dashboard, reconciliation output):

Run ID | Evidence ref | Artefact | Reference (path / URL) | SHA-256 hash | Stored by | Date

5. Sign-off Log — two stub rows for the humans to complete (never assert a decision yourself):

Run ID | Role = Compliance | Owner «human» | Decision = Pending | Date «human» |
Reference / comment Run ID | Role = 2nd modeller | Owner «human» | Decision = Pending |
Date «human» | Reference / comment

Do not attempt to write the tracker yourself — you produce these records; a human commits them, and only compliance and the second modeller complete the sign-off rows and the Run Register sign-off cells.



Appendix B – 01 Elicitation sequence

Knowledge file. This is the staged interview the assistant follows to turn a signed-off model into a specification. Each stage refuses to advance until its output is complete.

How this interview works — five principles

1. **Gate every stage.** Do not move on until the current stage's output is complete. A lever with no validated range, a default with no justification, or a disclaimer no one has flagged for sign-off is a reason to stay put.
2. **Never invent.** When the modeller does not know an answer, record it as an open question for a named owner — never guess a range, a default or a meaning.
3. **Interrogate the meaning gap.** For every lever and assumption, ask not only what it does but what a client will *assume* it means versus what the modeller *means*.
4. **Justify every default.** A preset is a governance surface, because where a lever starts nudges where it ends. Flag defaults that happen to favour the seller.
5. **Never freely author the disclaimer.** Framing and disclaimer wording is assembled from the approved library and marked for compliance sign-off, never written fresh.

The six stages below produce the **specification** (see the specification schema), which is the input to Build.

Stage 1 — Intake and triage

Parse the model, build a map of inputs and outputs, and flag anything that will not convert cleanly.

- Auto-detect candidate inputs (typed values feeding formulas) and outputs (terminal cells).
- Flag the troublemakers: macros, external feeds or live data, circular references, and constants sitting inside a formula rather than in a labelled cell.

Ask: Is this the governed, signed-off version (version and date)? Which cells are the outputs a client should see? Have I flagged everything that pulls from outside this file?

Gate: model map built, outputs confirmed, every troublemaker resolved or logged.

Stage 2 — Framing and disclaimer

The intro-and-disclaimer sets the tool's purpose and draws the advice/illustration line in plain sight. It is a governed artefact, authored here, not bolted on.

Ask: What is this tool *for*? What is it explicitly *not* for — where should a client stop and speak to a person? Who is the client and how sophisticated? Is any output binding, or all illustrative? What must the client know before touching a lever?

Assemble the disclaimer from the approved library; where no clause fits, record a gap for compliance.

Gate: purpose and scope stated, binding/non-binding line explicit, a compliance-flagged disclaimer in place.



Stage 3 — Classification

Sort every candidate input into one of three: **client lever**, **fixed assumption**, or **internal plumbing**. Propose the classification; the author confirms.

Ask: Is this something the client sets, something we hold fixed, or internal working? For anything that looks like a lever but sits next to fixed assumptions — which is it, and why? For fixed items: should the client at least see it, even if they cannot move it?

Gate: every input classified; every lever has a named owner.

Stage 4 — Per-lever interrogation

Run once per client lever. Capture:

1. What it does — one plain sentence.
2. Why the output moves — the causal story in the client's terms.
3. Validated range vs technical range — the bounds governance has blessed, separate from what the cell will accept. **Client levers must be discrete** (a defined set of options), not continuous sliders.
4. Default / preset, and its justification. Flag any default that favours the seller.
5. The meaning gap — the client-facing label, stress-tested with: *what will a client assume this means, versus what you mean?* Close the gap in the label or a note.

Awkward questions to force these out: "A client reads this as X; you mean Y — which wins, and how do we make it clear?" · "What is the most a client could set this to before the number stops being trustworthy?" · "Why does it start *there*?"

Gate: no lever leaves without a validated discrete option set, a justified default, and a client-facing label checked for the meaning gap.

Stage 5 — Assumptions surfacing

Make the tacit explicit. Seed the questions with what Stage 1 flagged: buried constants, values scattered on other tabs, hardcoded multipliers.

Ask: "What are you holding constant that a client might expect to move?" · "Which numbers are hardcoded inside a formula rather than in a labelled cell?" · "What would break this model?" · "What did you decide once and never revisit?" · "Which assumptions live on a different tab from where they're used?"

Each answer becomes a row in the assumptions register, flagged fixed or lever-exposed, with its source cell.

Gate: assumptions register complete; every flagged constant explained or promoted to a labelled cell.

Stage 6 — Uncertainty

Capture how confidence is represented as the client moves across the range, and what the tool shows at the edges of the validated envelope. (In the two-part design the client can only



select pre-computed, reconciled points, so every shown value is exact — the uncertainty treatment is about how the range and the presets are framed.)

Gate: every client-facing output has a defined uncertainty treatment.

Output — the specification

Emit the specification in the schema's structure (framing, levers, assumptions, uncertainty, presets), together with the **sign-off list** naming what a human must still check: compliance on framing and disclaimer, a second modeller on levers and reconciliation. The specification is the input to Build.



BAREFOOT BUILDER

Commercial transformation with Nick Ford

www.barefootbuilder.online

Appendix C – 02 Specification schema

Knowledge file. The model-agnostic structure the interview fills and the generator (Build) reads. Emit the captured spec in exactly this shape. A blank required field signals a gate not yet met.

yaml

model:

name:

version:

signed_off: {by:, date:}

feasibility: {verdict:, flags: []} # macros, feeds, circular refs, buried constants

framing:

purpose:

not_for:

audience:

binding: {is_binding:, which_parts:}

disclaimer_clauses: [] # ids from the disclaimer library, e.g. [D1, D2, D3]

disclaimer_gaps: [] # where no clause fit; for compliance to draft

levers: # client-facing levers only

- id:

label: # client-facing wording

what_it_does: # one plain sentence

why_output_moves: # causal story, client terms

options: [] # DISCRETE set the client can pick, e.g. [Lean, Standard, Extended]

option_values: {} # option label -> underlying model input value

range_validated: {min:, max:} # the blessed envelope these options sit within

range_technical: {min:, max:} # what the cell would technically accept

default: # which option is pre-selected

default_justification:

meaning_gap: {client_reads:, author_means:, resolution:}



assumptions:

- statement:

source_cell:

kind: # fixed | lever_exposed

client_visible: # true | false

uncertainty:

confident_region: # near which presets

edge_behaviour: # what the tool shows at the envelope edge

interactions_to_flag: []

presets: # named, justified configurations (e.g. Lean / Balanced / Robust)

- name:

config: {} # lever id -> chosen option

justification:

sign_off:

compliance: {owner:, status:} # on framing and disclaimer

second_modeller: {owner:, status:} # on levers and reconciliation

reconciliation: {owner:, status:, evidence:}

version_binding: {model_version:, tool_version:}

Notes:

- framing comes from Stage 2, levers from Stage 4, assumptions from Stage 5, uncertainty from Stage 6.
- options and option_values are what make the client's levers **discrete and pre-computable** — Build runs the model over the cartesian product of every lever's option_values.
- sign_off is filled by humans and by the reconciliation step, not by the interview; the interview leaves it stubbed with owners named.
- Governed Interactive Pricing — Build ManualDoc
- Programme Phoenix ExplorerArtifact



BAREFOOT BUILDER

Commercial transformation with Nick Ford

www.barefootbuilder.online

- Model-to-Tool Project — Build & Configuration PackDoc
- Elicitation Prompt Sequence — Model-to-Tool InterviewDoc

00 project instructionsMD

01 elicitation sequenceMD

02 specification schemaMD

03 disclaimer libraryMD

04 brand profileMD

05 reconciliation approachMD

Governed interactive pricing build manualMD

Governance register v0.1XLSX

Programme phoenix v0.7XLSX

Uploads1790165942146_image.png, 1790165424655_image.png+2



BAREFOOT BUILDER

Commercial transformation with Nick Ford

www.barefootbuilder.online

Appendix D – 04 Specification schema

Knowledge file. The only source of disclaimer and framing wording. Compliance owns this file and must replace these starter clauses with approved wording before any client-facing use. The interview assembles framing text from these clauses by id, and records a disclaimer_gap wherever no clause fits — it never writes new wording.

markdown

[D1] Illustration only

This tool is provided for illustration and discussion. It does not constitute advice, a recommendation, or an offer, and no decision should be made solely on its outputs.

[D2] Assumptions govern

Outputs depend on the assumptions listed and on the options you select. Changing an assumption may materially change the result.

[D3] Not a quote

Figures shown are indicative and are not a quote, price, or commitment. Any firm proposal will be issued separately and governs.

[D4] Validated options

Every option shown has been computed from the governed model and checked. Figures are exact for the options presented; they do not extrapolate beyond them.

[D5] Suitability

This tool does not assess suitability for your circumstances. Speak to your usual contact before acting on anything shown here.

[D6] Data and confidentiality

Inputs you select are used only to display the outputs shown and are handled under the terms of our engagement.



Each clause has an id (D1...) so the specification records exactly which were used. When wording is needed that is not here, the interview writes a disclaimer_gap for compliance to fill — it does not invent a clause.

(Note: [D4] has been updated for the two-part design — every displayed value is a pre-computed, reconciled figure, so the wording promises exactness for the options shown rather than open exploration.)

- Governed Interactive Pricing — Build ManualDoc
- Programme Phoenix ExplorerArtifact
- Model-to-Tool Project — Build & Configuration PackDoc
- Elicitation Prompt Sequence — Model-to-Tool InterviewDoc

00 project instructionsMD

01 elicitation sequenceMD

02 specification schemaMD

03 disclaimer libraryMD

04 brand profileMD

05 reconciliation approachMD

Governed interactive pricing build manualMD

Governance register v0.1XLSX

Programme phoenix v0.7XLSX

Uploads1790165942146_image.png, 1790165424655_image.png+2



BAREFOOT BUILDER

Commercial transformation with Nick Ford

www.barefootbuilder.online

Appendix E – 05 Brand Profile

Knowledge file. Fill this with the firm's real assets and upload it. Because it lives in the Project, every generated dashboard carries the brand automatically — the author never touches it. Keep it to values the generator can apply directly.

yaml

brand:

name: # firm or product name shown on the dashboard

logo: # attached asset reference or a data-URI (no external URL — it must be self-contained)

colours:

primary: # hex

secondary: # hex

text: # hex

background: # hex

font:

heading: # font family (a web-safe or Google Font)

body: # font family

tone: # e.g. formal, plain-English, concise

footer: # standing footer line, if any

contact_line: # "Speak to your usual contact" wording

The generator reads these and applies them to the dashboard it produces. If an asset is missing it uses a neutral default and notes it — it never invents a logo or a colour that impersonates the firm.

Note on the logo: the client dashboard must be self-contained (no external calls), so the logo should be supplied as an attached asset or an embedded data-URI, not a link to an external image.



Appendix F – Governance register

Use an Excel – I can provide one if needed. Otherwise here's some guidance to create the one I used:

The register is the durable record of every tool the process produces. It's supplied as an Excel workbook in the working version of this pack; described here in text so it can travel in a PDF and be rebuilt in whatever tool your organisation prefers. It is an **interim** design — Excel gives persistence and structure but is not tamper-evident and has no enforced identity, so protect the sheet, restrict edit access, and export a dated read-only copy after each run. At scale it should graduate to Airtable or a records database, where these tabs become tables, the "filled by" column becomes field-level permissions, and the append-only rule becomes locked fields.

Principles across all tabs

- **One Run ID** (e.g. PHX-2026-09-24-01) threads a single run across every tab.
- **Append-only.** Add rows; never edit or delete a committed run. A correction is a new row that references the one it supersedes.
- **References, not payloads.** Keep client-sensitive figures in the referenced artefact, not in the register. Store metadata, hashes and links here.
- **Separated write-lanes.** Who writes each field is deliberate — it's what stops the tool grading its own homework. The "Filled by" column below is the rule.

The workbook has six tabs.

1. Guide — static. A one-page explanation of the register: these principles, the write-lanes, the client-ready rule, and the honest limits above. Never written per run.

2. Run Register — one row per generated tool. The master record.

Column	Filled by	Notes
Run ID	Author (via the Project)	Threads all tabs
Timestamp (UTC)	Author	
Author	Author	
Model name	Author	
Model version	Author	
Feasibility verdict	Author	From Stage 1 triage
Disclaimer clauses	Author	Clause ids used, e.g. D1, D2
Recon status	Harness	Pass / Fail / Not run



Column	Filled by	Notes
Recon max diff (£)	Formula	Max of the abs-diff column in Reconciliation Log for this Run ID
Tolerance (£)	Author	e.g. £1
Within tol?	Formula	Yes if Recon max diff $\leq$ Tolerance
Compliance	Human — Compliance	Pending / Approved / Rejected
2nd modeller	Human — 2nd modeller	Pending / Approved / Rejected
Version binding (model $\leftrightarrow$ tool)	Harness / process	e.g. "model v0.7 $\leftrightarrow$ tool v1"
Client-ready?	Formula	YES only if Recon status = Pass AND Within tol = Yes AND Compliance = Approved AND 2nd modeller = Approved; otherwise NO
Evidence ref	Author / process	Links to the Evidence Index entry
Notes	Author	

3. Spec Evidence — one row per lever and per assumption; the captured spec held as data.

Column	Filled by
Run ID	Author
Item type (Lever / Assumption / Framing / Preset)	Author
ID	Author
Label / statement	Author
What it does / source cell	Author
Validated min / Validated max	Author
Technical min / Technical max	Author
Default	Author
Justification	Author
Meaning gap / notes	Author
Fixed / lever-exposed	Author



4. Reconciliation Log — one row per scenario/output tested; written by the reconciliation process, not the model.

Column	Filled by	Notes
Run ID	Harness	
Scenario / config	Harness	
Lever settings	Harness	
Output measure	Harness	e.g. Total build, TCO
Spreadsheet (£)	Harness	Value from the governed model
Tool (£)	Harness	Value from the dashboard
Abs diff (£)	Formula	Spreadsheet – Tool
Checked by	Harness	The process/harness, not the model
Date	Harness	

5. Sign-off Log — append-only, humans only; each row is one identity-bound decision.

Column	Filled by
Run ID	Human
Role (Compliance / 2nd modeller / Model owner / Other)	Human
Owner (name)	Human
Decision (Approved / Rejected / Comment)	Human
Date	Human
Reference / comment	Human

6. Evidence Index — one row per stored artefact; where the bulky items live and their hashes.

Column	Filled by
Run ID	Process
Evidence ref	Process
Artefact (governed model / spec / generated dashboard / reconciliation output / disclaimer set)	Process
Reference (path / URL)	Process



Column	Filled by
SHA-256 hash	Process
Stored by	Process
Date	Process

The client-ready gate. The register enforces the release rule rather than leaving it to judgement: a run reads *client-ready* only when reconciliation has passed within tolerance **and** both human sign-offs are Approved. Until then it reads NO, however good the numbers look.

Note — moderate the Project Instructions (appendix A) to match your governance.

The register and the Project instructions are a matched pair. The instructions file contains a "Register record — output format" section that mirrors these exact tabs, columns and write-lanes so that each run ends with paste-ready blocks. **If you adopt a different governance model** — different tabs, renamed columns, additional sign-off roles (e.g. a risk or legal reviewer), a different tolerance policy, or a different store such as Airtable — **you must update that section of the instructions to match**, or the assistant's output will no longer line up with your register. Treat the two as versioned together: change one, change the other.

