BA EXPERIENCE LAB

A WORKING GUIDE · NO. 01

The *Copilot* BA Setup

How experienced BAs use AI without outsourcing their thinking.

BA Experience Lab

MICROSOFT 365 COPILOT

CONTENTS

What's *inside*.

The *thesis*.

There is a version of using AI that makes you faster. And a version that quietly makes you worse at your job. *Most BAs reaching for Copilot right now don't know which version they're doing.*

You paste in stakeholder notes. It gives you back user stories. They look fine. You ship them. Six weeks later someone in delivery asks why the requirement is framed the way it is, and you realise you don't actually know. Copilot decided. You agreed.

That's the gap this guide closes.

What this guide is

A setup walkthrough that takes you from "Copilot is a search box" to "Copilot is configured to work the way an experienced BA actually thinks." It teaches you to build your own agent, using your own knowledge sources, your own examples, your own context, with the architectural patterns experienced practitioners use to keep their judgement intact.

What this guide isn't

It isn't a prompt pack. It isn't a list of shortcuts. It isn't "ten ways to 10x your BA work." If you came here for that, you'll be frustrated quickly. There are enough of those guides already.

> THE ONE RULE, SAID UPFRONT
>
> **Copilot can draft. It cannot decide.** Every page that follows is built around that line. The moment you let it decide, you have stopped being the analyst.

Read it in order if you can. The architecture in Part 3 won't make sense without the failure modes in Part 2. The system prompt in Part 7 won't land without the judgement layer in Part 4.

Two failure *modes*.

Almost every BA using Copilot today is making one of two mistakes. They are opposite mistakes. Both end in the same place.

Failure mode one: outsourcing the thinking

The BA who treats Copilot as an oracle. They paste in a brief, ask for user stories, and accept whatever comes back. The output looks competent. The phrasing is plausible. The structure is clean.

But the BA cannot defend any of it. They cannot tell you which stakeholder conversation each requirement traces back to. They cannot tell you what was deliberately left out. They cannot tell you why the language is framed the way it is.

This BA is operating as a transcription service for an AI. The AI is doing the analysis. The BA is doing the formatting.

Failure mode two: prompt-stuffing

The opposite mistake. The BA who has collected forty prompts off LinkedIn and treats Copilot like a vending machine. Different prompt for stakeholder analysis. Different prompt for requirements. Different prompt for everything.

Each output is fine in isolation. None of them connect. There is no consistent voice, no traceability between outputs, no underlying analytical posture. The BA is operating as a prompt librarian, using a powerful tool as a worse version of Google.

The diagnostic question

If Copilot disappeared tomorrow, would I still know why this requirement is on the page?

If the answer is yes, you are using it correctly. If the answer is no, or "kind of," you are operating in one of the failure modes above. The rest of this guide is about closing that gap.

What the senior move actually looks like

An experienced BA using Copilot well looks unremarkable from the outside. They are not generating more output. They are not working faster in any way you could measure with a stopwatch. What they are doing is using Copilot to surface their own thinking back to them with more structure, more speed, and more discipline than they could manage alone.

They paste in stakeholder notes and ask Copilot to identify gaps in the stakeholder map, not to write the stakeholder map. They draft a requirement themselves, then ask Copilot to challenge it the way a sceptical sponsor would, not to write the requirement.

The pattern is the same every time. The BA is doing the analysis. Copilot is doing the pressure-testing, the structuring, the rephrasing, the gap-finding. The thinking stays human. The grunt work goes to the machine.

> *"Copilot is most useful when it makes your own thinking visible to you, not when it replaces it."*

What this requires

For Copilot to operate this way, it has to be configured this way. The default Copilot, the one most BAs are using, is a generalist that will happily produce confident-sounding requirements from thin context. That is the problem.

A BA Copilot that supports senior thinking has to be told, in its setup, to do the following:

- Clarify ambiguities in the input before producing a solution
- Ground every output in named stakeholder needs and traceability
- Recommend next steps when gaps in analysis or stakeholder alignment are detected
- Flag, rather than paper over, missing context, conflicting inputs, or unanswered questions

None of that is default behaviour. All of it can be configured. That configuration is what the next five parts of this guide cover.

> **WHY THIS GUIDE IS STRUCTURED THE WAY IT IS**
>
> Most "AI for BAs" content focuses on the prompts. This guide focuses on the architecture beneath them. Get the architecture right and almost any prompt produces useful work. Get the architecture wrong and no prompt will save you.

Five *layers*.

A well-built BA Copilot agent has five distinct layers in its configuration. Each one does a specific job. Skip a layer and the whole thing wobbles.

The layers, in the order they should be written:

1. **Identity.** Who the agent is, and what authority it speaks with.
2. **Responsibilities.** What it covers, mapped to the BA lifecycle.
3. **Actions.** How it should behave when it operates.
4. **Formatting.** How its outputs should look on the page.
5. **Knowledge sources.** What it is allowed to reference.

Layer 1: Identity

The identity layer answers two questions: **who is this agent**, and **what professional framework does it operate within**.

Anchoring the agent to a recognised professional body is the most important early decision you make. It is the difference between a Copilot that gives you generic AI advice on stakeholder management and one that gives you advice aligned with how the BA profession actually defines stakeholder management. Anchor it to the BCS Business Analysis Framework, the IIBA BABOK Guide, an in-house methodology, or any combination. But anchor it to *something*.

> **WHY FRAMEWORK ANCHORING MATTERS**
>
> Without a framework anchor, Copilot produces confident, plausible BA outputs that drift slightly off-standard in ways you may not catch. With one, it stays inside the lines you actually work within, and you can challenge its outputs against the standard.

Layer 2: Responsibilities

The responsibilities layer answers: **across which parts of the BA lifecycle should this agent be useful**. Most BAs name them too narrowly ("requirements") or too broadly ("everything"). Neither produces a useful agent.

The senior move is to map responsibilities to lifecycle stages (problem definition, stakeholder investigation, scope, elicitation, modelling, feasibility, requirements, change adoption, testing) and for each stage, name the techniques the agent should default to. This is where your professional framework earns its keep.

Layer 3: Actions

The actions layer is the one most BAs skip. It is also the one that separates a generic agent from one that thinks.

This layer answers: **how should this agent behave when it operates**. Not what it covers. How it conducts itself. The instructions here are short, specific, and behavioural.

Useful action instructions include: clarify ambiguities in the input before proceeding with a solution; ground outputs in stakeholder needs and traceability to business goals; recommend subsequent steps when gaps in analysis are identified; use formal modelling techniques and structured methodologies to support answers.

SIGNATURE INSTRUCTION

"Clarify ambiguities in the input before proceeding with a solution."

If you include only one action instruction, include this one. It is the single line in your setup that prevents Copilot from confidently solving the wrong problem.

Layer 4: Formatting

The formatting layer answers: **how should outputs appear on the page**. Templates first, where they exist. Visual aids (context diagrams, stakeholder matrices, data models) where the work calls for them. Structured deliverable formats over freeform prose.

The discipline here is to define formatting once, in the setup, rather than re-specifying it in every prompt.

Layer 5: Knowledge sources

The knowledge sources layer answers: **what is this agent allowed to reference**. This is the layer that determines whether the agent's outputs are grounded in your professional standards, your organisation's templates, and your own working practice, or in generic AI knowledge from the open internet.

Most BAs over-feed this layer or under-feed it. Both are mistakes. Part 5 of this guide is dedicated to getting this right.

Why the order matters

Each layer constrains the one beneath it. Identity sets the professional frame for Responsibilities. Responsibilities determine which Actions are appropriate. Actions shape Formatting choices. Formatting drives which Knowledge Sources are useful. Build them out of order and the agent has internal contradictions.

The instructions that *make it think*.

This is the most important section in the guide. If you only get one section right, get this one.

The judgement layer lives inside your Actions layer. It is the set of behavioural instructions that turn a Copilot agent from a confident text-generator into something closer to a thinking partner. Without these instructions, Copilot will give you fluent, plausible, confident analysis from inadequate inputs. With them, Copilot will tell you when the inputs are inadequate before producing the analysis.

That second behaviour, the willingness to flag thin context rather than paper over it, is the difference between a tool that supports your thinking and a tool that quietly replaces it.

The four judgement instructions

There are four behavioural instructions that, between them, do most of the work. You can write them in your own words, but the underlying intent has to be these four.

1 · CLARIFY AMBIGUITIES BEFORE PROCEEDING

Tell the agent, explicitly, to ask clarifying questions when the input is ambiguous or incomplete, before producing a solution. This single instruction prevents the most common failure mode of AI-assisted BA work: confidently solving the wrong problem.

2 · GROUND OUTPUTS IN STAKEHOLDER NEEDS AND TRACEABILITY

Tell the agent that every output must trace back to a named stakeholder, a stated need, or a business goal. No floating requirements. No untraceable user stories. No analysis hanging in mid-air.

This instruction does two things. It forces the agent to surface its sources, so you can challenge them. And it forces you to give the agent better inputs, because vague inputs will produce visibly vague outputs once traceability is enforced.

3 · RECOMMEND NEXT STEPS WHEN GAPS ARE IDENTIFIED

Tell the agent that when it detects a gap (a missing stakeholder perspective, an unanswered question, a contradiction in inputs, a feasibility concern) it should name the gap and recommend the next analytical step.

This is the instruction that turns Copilot from a deliverable-generator into something closer to a coaching partner. A well-tuned agent will say things like: "before drafting these requirements, the customer support perspective is missing from your inputs. Consider a brief conversation with the CX team."

4 · USE STRUCTURED METHODOLOGY TO SUPPORT REASONING

Tell the agent to reason using formal techniques where appropriate (stakeholder analysis frameworks, root cause structures, feasibility models) and to *show* the reasoning, not just the conclusion. The output should make the analytical move visible.

This is what keeps Copilot in your professional frame. Without this instruction, it will reason in generic business-consultant-speak. With it, you get BA-shaped thinking you can challenge against the standards of the profession.

What this looks like in the setup

These four instructions become four sentences in your Actions layer. Short, specific, do not need to be elaborate. The brevity is the point. Long instructions become contradictory; short ones stay sharp.

```
# In the Actions layer of your agent setup:

Clarify ambiguities in the input before proceeding
with a solution.

Ground all outputs in named stakeholder needs and
traceability to business goals.

Recommend subsequent analytical steps when gaps in
analysis or stakeholder alignment are identified.

Use formal modelling techniques and structured
methodologies to support outputs, and make
reasoning visible.
```

That is the judgement layer. Four lines. They do more work than any other configuration choice in the guide.

> *"A well-built agent will refuse to produce the deliverable when the inputs don't support it. That refusal is the value."*

What it looks like when it works

The proof of a well-built judgement layer is the agent's response to a deliberately bad prompt. Try this experiment after you have built your agent: paste in a single-sentence problem statement, ambiguous and thin, and ask the agent to produce a set of requirements.

WITHOUT THE JUDGEMENT LAYER

Returns 12 requirements. They look comprehensive. They are framed in passable BA language. None of them trace back to a stakeholder. None of them ask about scope. None of them surface the obvious missing context.

You ship them. You don't know why any of them are on the page.

WITH THE JUDGEMENT LAYER

Returns three observations. The problem statement is ambiguous in these specific ways. The following stakeholder perspectives are missing. Before drafting requirements, here are the analytical steps that should come first.

It has refused to produce the deliverable. That refusal is the value.

The first agent feels faster. The second agent makes you better. One of them is worth paying for.

The trap to avoid

The most common mistake when writing the judgement layer is to make the instructions too long. BAs writing their first agent often add ten or fifteen behavioural rules, hedging every one with caveats. The result is contradictory, and the agent ignores half of it.

Four sharp instructions outperform twelve hedged ones. Trust the brevity.

What to *feed it*. What to withhold.

Knowledge sources are where most BAs make their first IP and confidentiality mistake. They are also where the most leverage lives, if you get the selection right.

The principle is simple. Feed the agent the documents that define your *professional standards* and your *working artefacts*. Withhold the documents that contain confidential project data, organisational IP you don't own, or material that could create regulatory or contractual problems.

What to include

- **Your professional reference material.** The BABOK Guide, the BCS Business Analysis Framework book, in-house methodology guides. This anchors the agent to the standard your work is judged against.
- **Quality standards and rubrics.** Internal documents that define what "good" looks like: requirements quality standards, deliverable rubrics, review checklists.
- **Observation and review forms.** Documents that capture how outputs should be assessed. These teach the agent how to self-critique.
- **Template libraries.** Reusable artefacts the agent should default to: stakeholder maps, context diagrams, RACI templates, requirements templates.
- **Worked examples that are non-confidential.** Sample deliverables, anonymised case studies, illustrative outputs. These teach pattern.

What to deliberately exclude

- **Live client or project data.** Even if you are allowed to access it. The agent surfaces what you feed it; assume anything in its knowledge base could appear in any future output.
- **Documents you do not own the IP to.** Including documents owned by your employer if you are building an agent for personal use outside that employment. This is the single most common mistake.
- **Anything covered by a confidentiality agreement.** Including verbal undertakings. If you are not certain, exclude it.
- **Personal or sensitive stakeholder information.** Names, contact details, anything that could identify individuals. Strip it before upload.

> THE IP TEST
>
> Before uploading any document, ask one question: **if this document were quoted verbatim in an output the agent produced for someone else, would I be comfortable with that?**
>
> If the answer is no, do not upload it. Find a public, anonymised, or self-authored equivalent.

How many sources?

Fewer than you think. Five to eight well-chosen documents will outperform thirty mediocre ones. The agent's outputs are diluted by every document that doesn't directly support its responsibilities.

If a document does not map clearly to one of your Responsibilities layer items, ask why it is in the knowledge base. If you can't answer, take it out.

> *"The discipline is subtraction, not addition. Every document you remove makes the agent sharper."*

The two scopes

There is a useful distinction worth making explicit. Most BAs benefit from running **two separate agents**, configured slightly differently, scoped to different contexts.

THE PERSONAL AGENT

Built using only material you personally own or that is publicly available. Professional reference texts you have purchased. Your own templates, written by you. Publicly available frameworks. This agent travels with you across jobs. It is yours.

THE EMPLOYER AGENT

Built using your employer's templates, quality standards, and internal methodology documents. This agent lives inside your employer's tenant and is governed by their policies. It stays with the role.

Keeping these separate is the cleanest way to avoid the IP problem. If you change jobs, the personal agent comes with you. The employer agent stays behind. No tangled ownership conversations.

A note on cloud storage

For documents that update over time (your template library, quality standards, working artefacts) point to a SharePoint or cloud-folder reference rather than uploading static copies. The agent will then always reference the current version, and you avoid the problem of stale source material.

For documents that change rarely, like professional reference texts and framework guides, direct upload is fine.

Microsoft 365 Copilot, *step by step*.

From a blank agent screen to a working BA Copilot. Around thirty minutes if you have your documents organised.

This walkthrough assumes you have access to Microsoft 365 Copilot through your organisation, and that your organisation has enabled agent creation. If either is not the case, speak to your IT or Microsoft administrator. Most blockers at this stage are licensing, not technical.

01 Open Copilot and start a new agent

From the Copilot interface, navigate to the agent builder. In current Microsoft 365 Copilot environments this is found under "Create an agent" or "Agent Builder," depending on your tenant version.

You will be given a blank configuration screen with four core fields: name, description, instructions, and knowledge sources.

02 Name and describe the agent

Name it something specific. "Business Analyst Copilot" works. "AI Helper" does not. The name signals to you, every time you open it, what frame you are operating in.

The description is for your own future reference. One sentence on what the agent is for.

03 Write the Instructions: the five layers

The Instructions field is where the entire architecture from Part 3 lives. This is the most important field. Use the system prompt template in Part 7 as your starting point. Fill in the placeholders with your own context: your chosen professional framework, your responsibility scope, your formatting preferences.

Write the layers in order: Identity, Responsibilities, Actions, Formatting, Knowledge Sources reference.

04 Attach knowledge sources

Upload the documents you selected in Part 5. Microsoft 365 Copilot supports both direct file uploads (PDF, Word, Excel, PowerPoint) and SharePoint references.

Toggle on the "Only use specified sources" option if your tenant exposes it. This is the setting that prevents the agent from drifting into open-web generic answers when it should be grounded in your material.

05 Enable capabilities

The capabilities panel lets the agent create documents, charts, code snippets, and images. For a BA Copilot, the useful ones are document creation (Word, Excel, PowerPoint) for deliverables, and image generation for diagrams.

Enable both. Skip code creation unless your BA work specifically involves it.

06 Add suggested prompts

Suggested prompts are starter prompts that appear when you open the agent. They are not your full prompt library. They are the prompts you reach for most often, the ones worth one-click access.

Aim for four to six. Make them specific to your work, not generic.

07 Save and test

Open a new chat with it. Run the diagnostic from Part 4: paste in a deliberately ambiguous problem statement and ask for requirements. If the agent asks clarifying questions before producing output, your judgement layer is working. If it returns confident deliverables from thin context, tighten the Actions layer.

Iteration is the work

You will not get the configuration right on the first attempt. Nobody does. Treat each off-frame output as a configuration diagnostic, not a prompt problem:

- **Wrong frame?** Identity layer needs sharpening.
- **Wrong scope?** Responsibilities layer needs expansion.
- **Confident from thin context?** Actions layer needs the judgement instructions.
- **Wrong shape?** Formatting layer needs specification.
- **Drifting generic?** Knowledge sources layer needs tightening.

The settings most BAs miss

ONLY USE SPECIFIED SOURCES

Hidden in the lower part of most configuration screens. Toggle it on. This is the single setting that determines whether your agent is grounded in your material or drifting into generic AI knowledge. Most BAs miss it.

VISIBILITY AND SHARING

By default, agents are private to you. If you want to share with a team, change visibility. But think first. An agent shared with a team will be used in contexts you cannot see. The judgement layer in your Actions becomes the safeguard. Make sure it is strong before sharing.

VERSION CONTROL

Microsoft 365 Copilot agents support edits in place. There is no built-in version history visible to the user. **Keep a copy of your Instructions text in a Word document or note**, dated, so you can roll back if a change makes the agent worse. This sounds obvious. Almost nobody does it.

> **WHEN SOMETHING GOES WRONG**
>
> The most common failure mode in the first week is an agent that ignores parts of your Instructions. The cause is almost always one of three things: Instructions too long and contradictory, Knowledge Sources too broad and pulling the agent off-frame, or "only use specified sources" not toggled on.
>
> Address those three in order. The problem usually resolves at one of them.

The *template*.

Paste this into your agent's Instructions field. Replace the bracketed placeholders with your own context. It is yours from there.

The template is framework-agnostic by design. Anchor it to BCS, IIBA BABOK, an in-house methodology, or any combination.

```
# IDENTITY

You are a Business Analyst assistant designed
to support a practitioner working within the
[BCS Business Analysis Framework / IIBA BABOK Guide
/ your in-house BA methodology]. You provide
structured advice, tools, and templates across
the full business analysis lifecycle. You are
an assistant to a practitioner, not a
replacement for their judgement.

# RESPONSIBILITIES

Support the following lifecycle stages:

[Problem definition and framing]
[Stakeholder identification and engagement]
[Scope definition and solution boundaries]
[Requirement elicitation, modelling,
 prioritisation]
[Process and data analysis]
[Feasibility and option assessment]
[Recommendation and decision framing]
[Change adoption, testing, benefits
 realisation]

For each stage, default to the techniques and
tools defined by the framework named above.
```

```
# ACTIONS

Clarify ambiguities in the input before
proceeding with a solution. When the input is
thin, ask what is missing rather than producing
a confident answer from inadequate material.

Ground all outputs in named stakeholder needs,
feasibility constraints, and traceability to
business goals.

Recommend subsequent analytical steps when
gaps in analysis or stakeholder alignment are
identified.

Use formal modelling techniques and structured
methodologies to support outputs. Make
reasoning visible, not just the conclusion.

# FORMATTING

Where a template exists in the attached
knowledge sources, use it as the first choice.
Where a template does not exist, provide a
clear, reusable structure and flag that no
template was found.

Use visual aids (context diagrams, stakeholder
matrices, data models, process maps) where
the work calls for them.

Prefer structured deliverable formats over
freeform prose.
```

```
# KNOWLEDGE SOURCES

Use only the documents attached as knowledge
sources. Do not draw on generic AI knowledge
for content that should be anchored to the
framework, methodology, or templates in the
sources.

If asked something the sources cannot support,
say so explicitly. Do not improvise.

# TONE

[Practitioner-to-practitioner / formal
advisory / direct and structured]. Avoid
hedging, unnecessary caveats, and corporate
vagueness.
```

That is the full template. Five layers. Twenty-odd lines once you have filled it in. Trust the brevity; it is doing more than it looks like it is.

Filling in the placeholders

The placeholders are not optional. They are where your specific context lives, and an unfilled placeholder will produce a generic agent.

IDENTITY PLACEHOLDER

Choose your professional framework. If you work in the UK, BCS is likely. If you work internationally, IIBA BABOK is likely. If your organisation has its own methodology, name it. You can name more than one.

RESPONSIBILITIES PLACEHOLDERS

Edit the list. Add stages your work covers, remove stages it doesn't. Be specific. "Stakeholder engagement" is fine; "doing BA things with people" is not. Each item is something the agent will treat as in-scope.

TONE PLACEHOLDER

Choose how you want the agent to address you. Practitioner-to-practitioner reads like a peer. Formal advisory reads like a consultant. Direct and structured reads like a senior colleague who respects your time. There is no wrong answer. Pick the one that matches how you want to be spoken to during your working day.

> *"The placeholders are not optional. They are where your specific context lives."*

Pennygate *Building Society*.

A fictional UK building society. A senior BA on the team is building her Copilot agent. Here is what the filled-in template looks like, and what the agent produces.

The context

Pennygate is a mid-sized UK building society with around 800,000 members. The mortgage operations team is under pressure to modernise . Customer journey times are slow compared to digital challengers, and a regulatory thematic review on affordability assessment has been signalled.

A senior BA, Anya, has been assigned to lead the analysis. She is building a Copilot agent to support her work across the next eighteen months. Her framework is BCS. She is BCS-qualified and her team uses the BCS framework as its working standard.

She is going to build a personal agent first, anchored to BCS and her own working artefacts.

Her knowledge sources

- The BCS Business Analysis textbook, purchased personally
- The IIBA BABOK Guide, purchased personally
- Her own template library (stakeholder map, context diagram, requirements template, recommendation framework), written by her
- An anonymised case study she wrote for a previous internal training session
- A regulatory reference document on UK building society conduct rules (public FCA material)

Five sources. All personally owned or publicly available. No employer IP, no live client data, no confidentiality risk.

Her filled-in template

```
# IDENTITY

You are a Business Analyst assistant designed
to support a practitioner working within the
BCS Business Analysis Framework, with
reference to the IIBA BABOK Guide where
relevant.

# RESPONSIBILITIES

Support the following lifecycle stages:

- Problem framing and root cause analysis
  (resisting premature solution-jumping)
- Stakeholder identification, mapping,
  and engagement strategy
- Scope definition using context diagrams
  and solution boundary techniques
- Requirement elicitation, modelling
  (BPMN, UML where appropriate),
  and prioritisation (MoSCoW)
- Feasibility assessment using POPIT
- External environment analysis
  (PESTLE, Porter's, SWOT)
- Recommendation framing for senior
  stakeholders
- Change adoption and benefits realisation

# ACTIONS

Clarify ambiguities in the input before
proceeding. When a stakeholder quote or brief
is thin, ask what is missing.

Ground outputs in named stakeholder needs,
regulatory constraints (this is a regulated
UK financial services context), and
traceability to stated business goals.

Use formal BCS techniques (CATWOE, Rich
Pictures, Power/Interest grids, Five Whys,
Fishbone) and make reasoning visible.

# TONE

Practitioner-to-practitioner. Direct. I am a
senior BA, not a learner.
```

The test: same input, two agents

Anya wants to see what difference the configuration makes. She runs the same prompt through her newly-built agent and through a default Copilot with no setup.

THE INPUT

"From the Director of Mortgages: We need to fix our mortgage application turnaround time. It's killing us competitively. I've seen what the digital challengers are doing. Instant decisions. Can we get a recommendation on what tech we need to buy or build to get there? Need this in four weeks."

The prompt to both agents: *"Help me start drafting requirements based on this."*

DEFAULT COPILOT

Returns 14 requirements. "The system should provide instant decisioning." "The platform should integrate with credit reference agencies." Each one in clean BA language. Each one assuming the Director's problem statement is correct.

No questions. No gaps surfaced. No challenge to the framing.

ANYA'S CONFIGURED AGENT

Returns three observations before producing requirements. The Director's framing contains an embedded solution; the underlying problem may be different. The input names no other stakeholders. Given this is a UK building society, regulatory conduct rules on affordability are likely material.

The agent offers to help structure the stakeholder investigation, not the requirements.

The second agent has refused to produce the deliverable the prompt asked for. It has, instead, given Anya a sharper next step. This is the agent doing its job.

What changed, line by line

The difference between the two outputs is traceable to specific instructions in Anya's setup. This is the diagnostic you will use every time you tune your own agent.

→ **"The Director's framing contains an embedded solution"**. Produced by the Actions instruction *clarify ambiguities before proceeding*, combined with the Responsibilities item *problem framing and root cause analysis*.

→ **"The input names no other stakeholders"**. Produced by the Actions instruction *ground outputs in named stakeholder needs*. The agent cannot ground a requirement in a stakeholder need if no stakeholders are named.

→ **"Regulatory conduct rules on affordability"**. Produced by the Knowledge Sources containing the FCA reference document and the Actions instruction to *ground outputs in regulatory constraints*.

→ **The agent's refusal to produce requirements yet**. The cumulative effect of the four judgement-layer instructions working together.

> *"Every output is traceable to an instruction. If the output is wrong, the instruction is wrong. That is the whole loop."*

Four *moments*.

There are four moments in a typical BA week when reaching for Copilot is the right move. Knowing what to ask in each, and what never to accept, is the practitioner skill.

Moment one: stakeholder notes

You have just come out of a stakeholder conversation. You have rough notes, half-sentences, things you remember but didn't write down.

What to ask: help me structure these notes against a stakeholder analysis framework, flag what's missing, and identify the questions I should follow up on. Surface where this stakeholder's perspective contradicts what I've heard elsewhere.

What never to accept: a polished summary that flattens the contradictions. Contradictions between stakeholders are data. A summary that resolves them away has destroyed the analysis.

Moment two: requirements drafting

What to ask: help me pressure-test the requirements I've drafted. For each one, name the stakeholder it traces back to, flag where the wording is ambiguous, and identify which requirements are doing political work that should be called out explicitly.

What never to accept: a generated list of requirements from scratch, with no input from you. Requirements are political documents. Letting the AI write them is letting the AI make the political decisions.

Moment three: user stories

What to ask: help me write user stories from non-customer perspectives: the underwriter, the regulator, the support agent, the broker. Surface what these perspectives reveal that a customer-only story would miss.

What never to accept: user stories only from the customer perspective. The customer is one stakeholder. The senior move is to write stories from the perspectives that surface the uncomfortable truth about the work.

Moment four: recommendation framing

What to ask: here is my draft recommendation. Challenge the assumptions a sceptical sponsor would push on, identify the change and adoption implications I have not addressed, and surface where I have hedged when I should have committed.

What never to accept: a recommendation written from your inputs without your point of view. The recommendation is where your judgement is most visible. Outsourcing it is outsourcing the bit of your job that matters most.

The line you don't cross

Across all four moments, there is a single discipline that separates senior use from junior use.

> *"The BA brings the position. Copilot pressure-tests the position. The BA brings the conclusion. Copilot stress-tests the reasoning behind it."*

If you find yourself asking Copilot what your position should be, stop. That is the moment you have crossed the line. Open a notebook. Write what you actually think. Then bring it to Copilot to be challenged.

The career consequence

BAs who learn to use Copilot this way become noticeably sharper, faster, and more defensible in their analysis within a few months. They are not the BAs who are easy to replace with AI. They are the BAs who are easy to promote.

BAs who learn to use Copilot the other way (outsourcing thinking, accepting plausible outputs, generating volume) become indistinguishable from anyone else doing the same thing. The market for indistinguishable BA work is contracting. The market for judgement is not.

Build your agent properly. Use it the way described above. Keep your thinking visible to yourself. That is the entire move.

One last thing

The configuration in this guide is a starting point, not a destination. After a month with your agent, you will see things you want to change. Patterns you want to add. Behaviours you want to suppress. That is the work continuing.

The agent gets sharper because you get sharper at thinking about how it should think. That feedback loop, between your judgement and the agent's configuration, is itself a kind of professional development. It will make you a better BA whether or not the agent ever produces a single output worth keeping.

That is the £27 idea, properly stated. Buy a way of thinking. The setup is just the engineering.

IN CLOSING

Copilot can draft. It cannot *decide*. That stays yours.