evaluagent

A WHITEPAPER FOR QUALITY AND CX LEADERS

How to *burst* the bubble of an unrealistic self-build promise

Ten promises that turn up in self-build AutoQA business cases, what tends to happen next, and the questions to ask before anyone commits.



For Heads of Quality and CX leaders weighing an in-house build against a specialist platform.

Complete visibility across every interaction — human and AI

INSIDE

Contents

Executive summary 3
The ten promises and the numbers behind them, on one page.

Who this is for and how to use it 4
Who the paper is written for, and how each chapter works.

You can build it. The question is whether you should. 5
What the demo shows, and what sits below the waterline.

1 "It'll be accurate." 6
Accuracy only means something once it's defined against your own team's judgement, and measured.

2 "Models are just an API call." 8
Providers retire and replace models regularly, and each change can move your scores without warning.

3 "It'll be cheaper." 10
Year one is the cheap year. The real comparison runs over three.

4 "We'll be live by Q2." 12
A conversation can be scored in a demo within days. Production takes most of a year.

5 "Our engineers have capacity." 14
Spare capacity looks free on paper. It comes out of your product roadmap.

6 "Security's covered." 16
QA touches every customer conversation, so its security bar is among the highest in your stack.

7 "Agents will trust the scores." 18
A score nobody can explain soon becomes an employee relations problem.

8 "We can judge our own AI agents." 20
As AI agents take on more conversations, someone other than their builders needs to judge them.

9 "If others can, we can." 22
Some QA vendors have moved their focus away from AutoQA. Doing it well is a full-time job.

10 "We'll know if it breaks." 24
AI fails silently. QA is meant to be the thing that notices.

The checklist 26
Ten questions: is the build case ready?

Next step, sources and assumptions 27
Book a session, and the figures and sources behind this paper.

IN BRIEF

Executive summary

A prototype that scores a customer conversation is now within reach of almost any team. The prototype is rarely where the trouble starts. The trouble is everything a quality programme needs around it once real agents and real customers are involved.

AT A GLANCE

£1.32m	£336k	Month 9–12	Week 4–6
to build in-house over three years	with **evaluagent** over the same three years	first value on an illustrative internal build plan	value from **evaluagent**

Illustrative figures for a 300-seat contact centre handling around 80,000 conversations a month. Sources and assumptions are on the back page.

Self-build business cases tend to rest on the same ten promises. The scores will be accurate. Models are just an API call. It'll be cheaper. We'll be live by Q2. Each one sounds reasonable on a slide, and each hides a cost that arrives later, usually after the decision has been made.

This paper takes the promises one at a time. For each, it sketches how the promise can play out, sets out the argument, and gives you questions to put to your team, your CTO or your CFO.

In an illustrative 300-seat contact centre, building costs around £1.32m over three years against around £336k with evaluagent. On an illustrative build plan, first value arrives around month nine to twelve. evaluagent delivers value from week four to six.

If your build case has a confident, costed answer to almost every question, it's a well-tested plan. If it doesn't, you'll know where to look.

BEFORE YOU READ ON

Who this is for and how to use it

This paper is written for Heads of Quality and CX leaders in contact centres who are being asked to back a plan to build AutoQA in-house, or to argue against one. It assumes you know your quality programme well and that the people across the table know engineering and budgets well. The aim is a view of what the build involves that both sides of the table can agree on.

Each chapter follows the same pattern, so you can go straight to the promise you're hearing.

1 **Find the promise you're hearing.**
Each chapter opens with one line from a self-build business case, and an illustrative timeline of how it can play out.

2 **Take its questions into the room.**
Every chapter ends with questions for your team, your CTO or your CFO.

3 **Count the "not yets".**
Every answer that isn't confident and costed is a "not yet". The checklist on page 26 keeps score.

Nothing here argues that your team couldn't build it. The question is what it would take to run well, for years, and whether that's the best use of the people involved.



Figure 1 The conversation this paper is for. A Head of Quality takes the build case to the CFO and CTO.

BEFORE YOU START

You can build it. The question is whether you *should*.

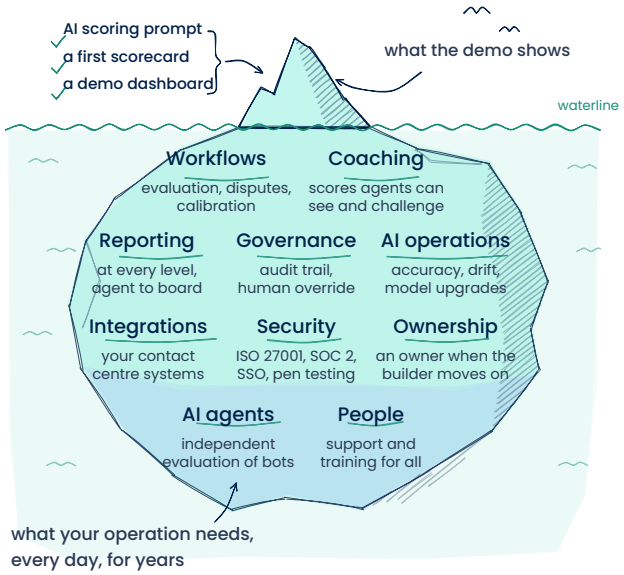


Figure 2 What the demo shows, and what your operation needs every day, for years.

> *Scoring a conversation is quick to demo. Running quality management is a permanent product commitment.*

With today's AI tools, almost any team can stand up a prototype that scores a customer conversation. That's the easy part, and it's the part that shows well in a demo.

Running quality management that agents trust, leaders act on and regulators accept is a different commitment. It's a permanent product, with an owner, a roadmap and a support load that doesn't end at launch.

Picture an iceberg. Above the waterline sit the AI scoring prompt, a first scorecard and a demo dashboard. Below it sits what your operation needs every day: workflows for evaluation, disputes and calibration, coaching from scores agents can see and challenge, reporting at every level, governance and an audit trail, accuracy and drift checks through every model upgrade, integrations, security, independent evaluation of AI agents, an owner for when the builder moves on, and the people to support and train everyone who uses it.

Most of the long-term value, and most of the investment, sits below the line. That's the part internal builds tend to underestimate, because none of it appears in the demo.

The ten chapters that follow each take one promise from a typical build case and look at what sits underneath it.

PROMISE 1 OF 10

1

“It’ll be accurate.”

Accuracy only means something once it’s defined against your own team’s judgement, and measured.

AN ILLUSTRATIVE TIMELINE

Week 2

The prototype scores 20 calls. They look great.

Month 3

A team lead disputes a score. Nobody can say why it was given.

Month 6

Agents stop trusting the scores. QA drifts back to spreadsheets.

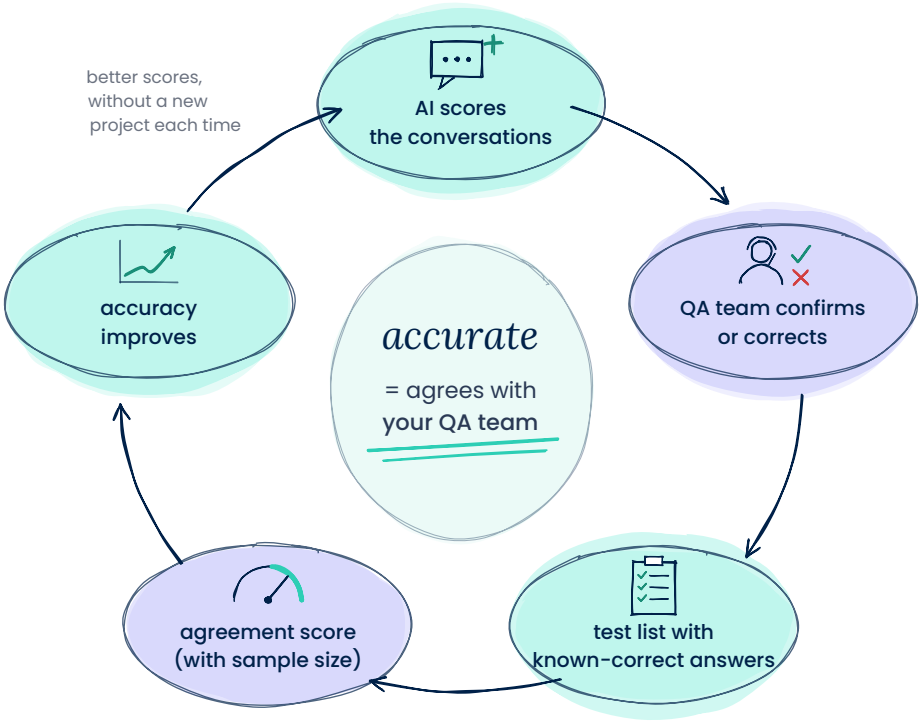


Figure 3 Accuracy is measured, not eyeballed. Analysts confirm or correct results, test lists hold known-correct answers, and agreement is reported with its sample size.

There's no universal definition of a well-handled call. Every contact centre has its own standards, its own regulatory context and its own view of what good looks like. So accurate can only mean one thing in practice. The AI agrees with your own QA team's judgement, often enough and consistently enough that you'd stand behind its scores.

Measuring that takes machinery. You need a workflow where QA analysts confirm or correct the AI's results. You need test lists of conversations with known-correct answers, so a change can be checked before it reaches anyone's scores. You need an agreement score with a sample size behind it, because a figure based on twenty calls tells you very little. And you need a way to improve accuracy with minimal extra effort, so each correction makes the next score better instead of starting a new project.

Most internal builds have none of this. The prototype is judged by people reading a handful of results and agreeing they seem sensible. That's a reasonable first test, but nobody can put a number on it.

The difference shows the first time someone challenges a score. "Does it look right?" and "is it accurate?" are different questions, and only one of them has an answer you can show.

> *A figure based on twenty calls tells you very little. Ask how many conversations sit behind the number.*

Questions to ask your team

1. What does "accurate" mean for us, and who decided?
2. How many conversations is the accuracy figure based on?
3. Who reviews and corrects the AI's results, and how often?
4. How does a correction make the next score better?

THE PROOF POINT

"Does it look right?" isn't an answer you can give a regulator, an ombudsman or an agent disputing their score.

PROMISE 2 OF 10

2

“Models are just an API call.”

Providers retire and replace models regularly, and each change can move your scores without warning.

AN ILLUSTRATIVE TIMELINE

Day 1
We build on the latest model. Scores look stable.

Month 7
The provider announces the model is being retired.

Month 8
Every scorecard needs re-testing before anyone trusts the scores again.

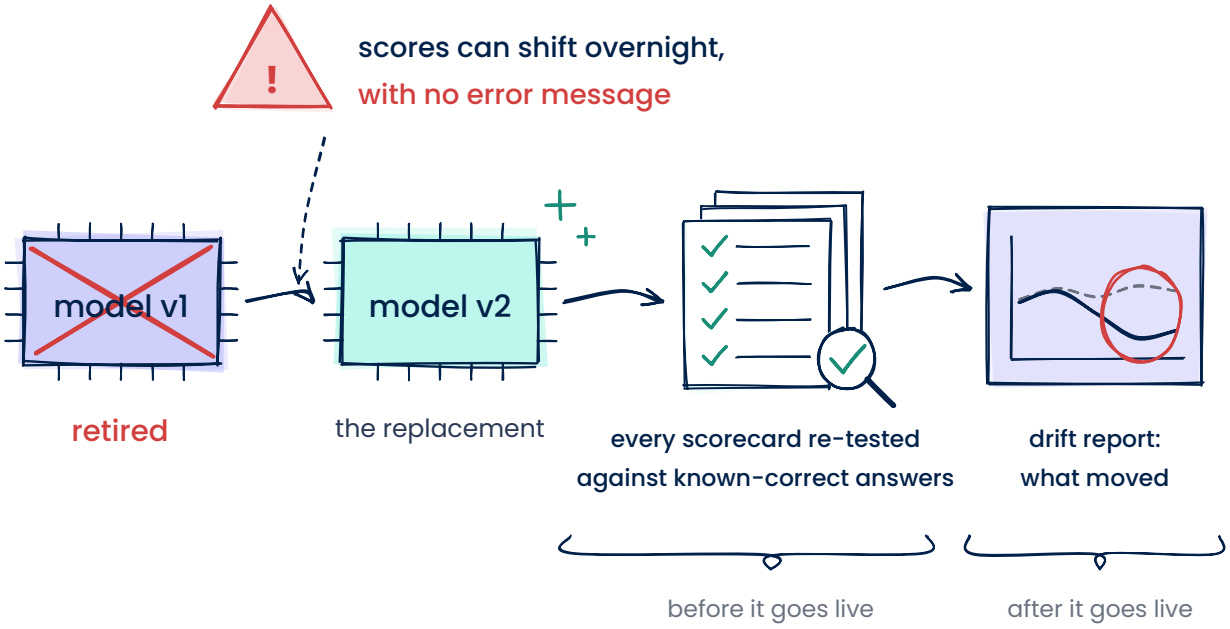


Figure 4 A model change arrives on the provider’s timetable. Every scorecard has to be re-tested before the scores can be trusted again.

Building on a large language model feels like building on a fixed foundation. It isn't. The model underneath has a shelf life set by someone else, and when it's replaced, scores can shift overnight. There's no error message when it happens. The system keeps producing scores. They just aren't the scores it would have produced last week.

For a quality programme that's a serious problem. Scores feed coaching conversations, performance reviews and trend reporting. If the model underneath changes, a dip in an agent's results might be about the agent, or it might be about the model. Without a way to tell the difference, nobody can trust the trend.

Keeping scores consistent means re-testing every scorecard against known-correct answers before a new model goes live, then watching for drift afterwards. If you run several scorecards across different teams or lines of business, that's several re-validations each time, and each one needs people who know what the right answer looks like.

This is a recurring cost, and the provider's release schedule sets the timing, not yours. It's missing from most build estimates, including the illustrative one in the next chapter.

> *"Was that dip about the agent, or about the model? Without change control, nobody can say.*

Questions to ask your team

1. Who re-validates every scorecard when the model changes?
2. How would we spot a silent shift in scores?
3. How long does one re-validation take, and what does it cost?
4. What happens to coaching and reporting while it's under way?

THE PROOF POINT

With evaluagent, model change control is our job. Every scorecard is re-tested before a change goes near your scores, and drift reporting shows what moved.

PROMISE 3 OF 10

3

“It’ll be cheaper.”

Year one is the cheap year. The real comparison runs over three.

AN ILLUSTRATIVE TIMELINE

The napkin

“One engineer, three months.”

Year 1

Inference bills grow with every conversation scored.

Year 2

The tool one engineer started now needs two.

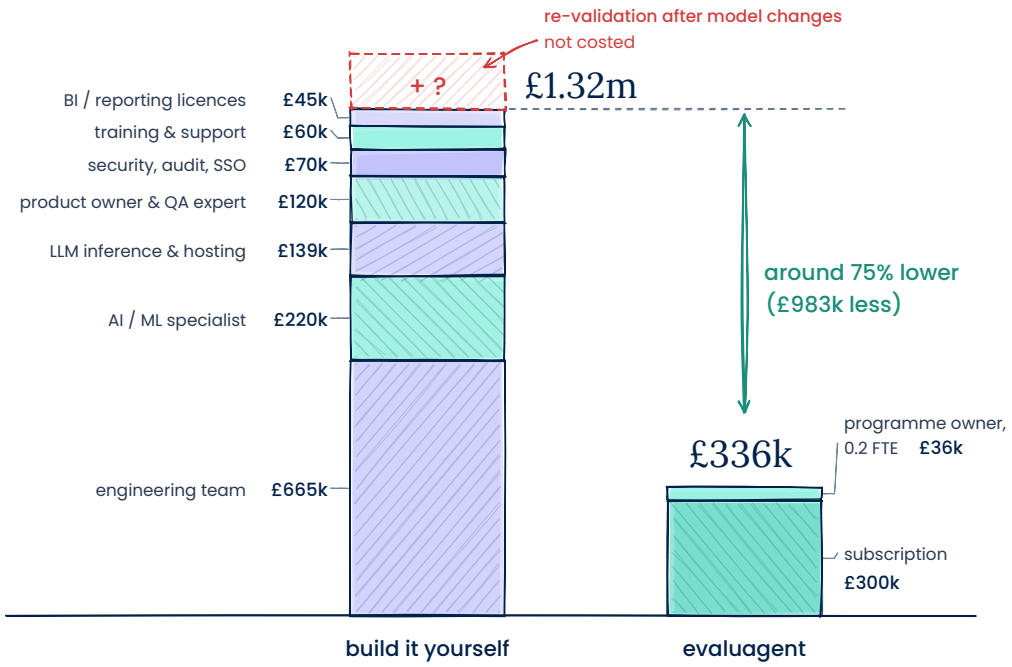


Figure 5 Illustrative three-year cost for a 300-seat contact centre. Build £1.32m, evaluagent £336k. Excludes transcription and re-validation after model changes.

Build estimates usually start on a napkin. One engineer, a few months, an API key. The first-year figure looks small because year one is the cheap year, and it rarely includes the people who'll run the thing once it's live.

Then usage grows. Every conversation scored is an inference call, so the bill climbs with volume and with prompt complexity. Each new team or line of business that wants its own scorecard adds prompts to write, test and maintain. Security testing, reporting licences, training and support arrive on top of the engineering. And the engineering doesn't stop at launch, because a live quality programme keeps asking for changes.

Over three years, most of the build cost is people: engineers, an AI specialist and a product owner who would otherwise be working on something else. Two things sit outside the figures below. The build total leaves out re-validation after model changes, covered in the previous chapter. Neither total counts the value lost while a build works its way to production, which is the subject of the next.

> *"Year one is the cheap year. Most of the build cost is people, and it keeps coming.*

Illustrative three-year cost of ownership

300-seat contact centre, around 80,000 conversations a month

BUILD IN-HOUSE	THREE-YEAR COST
Engineering team (3 FTE, then 2)	£665k
AI / ML specialist	£220k
LLM inference and hosting	£139k
Product owner and QA expert	£120k
Security, audit and SSO	£70k
Training and support	£60k
BI and reporting licences	£45k
Re-validation after model changes	*not costed*
Total	**£1.32m**

WITH evaluagent	THREE-YEAR COST
Subscription (illustrative, £100k a year)	£300k
Programme owner (0.2 FTE, £12k a year)	£36k
Total	**£336k**

Around 75% lower with evaluagent, a difference of roughly £983k. Assumes £95k fully loaded per engineer, £110k per AI specialist and LLM cost of around £0.04 per conversation. Excludes transcription, which both routes need. Your numbers will differ.

Questions to ask your team

1. Have we costed years two and three, not just year one?
2. Does the estimate include inference growth, security, reporting licences, training and support?
3. Does it include re-validation after model changes?
4. What does each month without full coverage cost us?

THE PROOF POINT

Illustrative three-year cost: £1.32m to build, £336k with evaluagent. Around 75% lower, before counting delayed value.

PROMISE 4 OF 10

4

“We’ll be live by Q2.”

A conversation can be scored in a demo within days. Production takes most of a year.

AN ILLUSTRATIVE TIMELINE

Q2
Requirements, design, the scoring engine and prompts.

Q3
Integrations, workflows and reporting. Still building.

Q4
Security sign-off. The pilot hasn’t started yet.

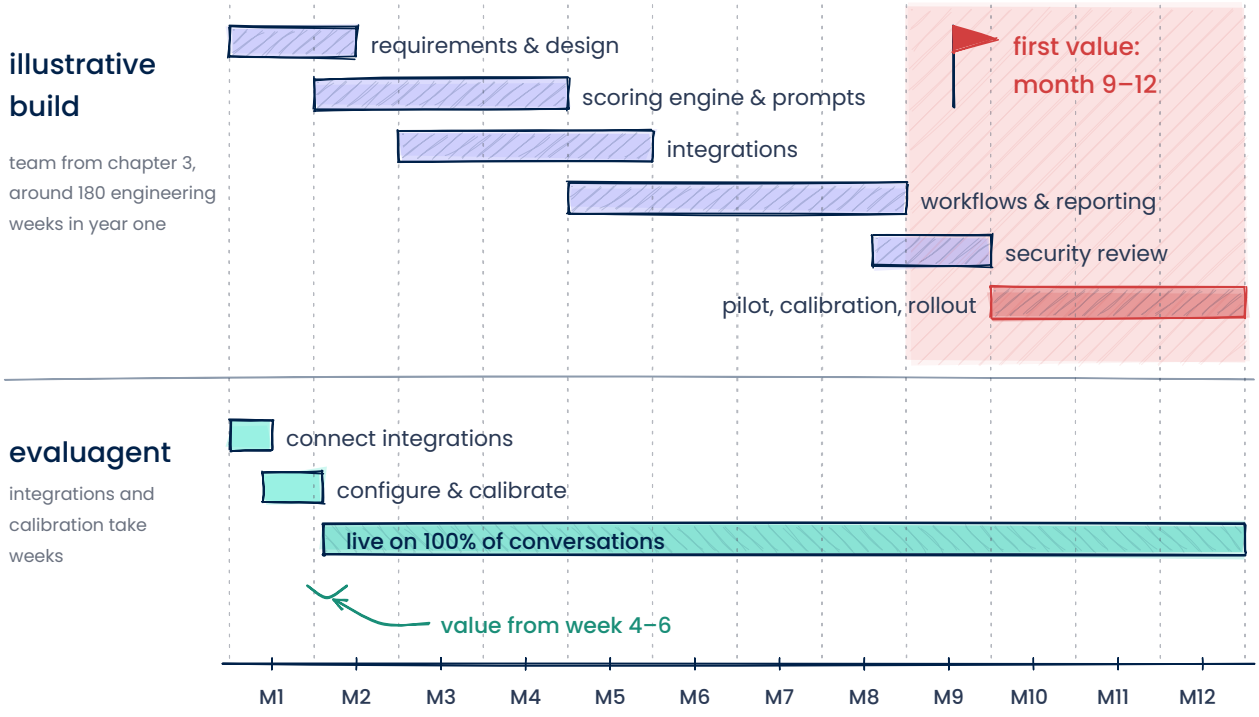


Figure 6 Illustrative route to first value, using the team costed in chapter 3. Durations and basis are set out overleaf.

A demo is the quickest part of any build, which is why the go-live date in a build plan is often optimistic. The demo covers one step of the journey. Production covers all of them.

Getting to production means working through every stage in the table below. Some of it runs in parallel. Much of it can't, because each stage depends on decisions made in the one before.

Workflows and reporting take longest because they aren't one screen. They're evaluation, dispute and calibration flows, coaching, and separate views for agents, team leads, QA analysts and compliance, each with its own permissions, plus reporting from a single agent up to the board.

These durations are an estimate, not a benchmark. They assume the team costed in chapter 3, three engineers and an AI specialist, which comes to around 180 engineering weeks in year one. If your plan is shorter, check which stages it leaves out.

With evaluagent the platform is already built, so the work is connecting it to your systems and calibrating it to your standards. That takes weeks, and value arrives from week four to six. The gap matters most when a renewal, a regulatory review or an AI agent rollout is on the calendar, because none of them will wait for the build.



Figure 7 Q2 becomes Q3, then Q4.

Illustrative route to first value

Built by the team costed in chapter 3

STAGE	DURATION
	BUILD IN-HOUSE
Requirements and design	6 weeks
Scoring engine and prompts	3 months
Integrations	3 months
Workflows and reporting	4 months
Security review	6 weeks
Pilot, calibration and rollout	*follows sign-off*
First value	**month 9–12**
	WITH evaluagent
Integrations and calibration	*weeks*
First value	**week 4–6**

An estimate, not a benchmark. Some stages run in parallel, so the durations don't add up to the total. Swap in your own team's figures.

Questions to ask your team

1. When will we be live on 100% of conversations, not a pilot?
2. What's on the critical path, and who owns each part?
3. What does each month of waiting cost us?
4. Is a renewal, regulatory review or AI agent rollout due before then?

THE PROOF POINT

Live in weeks, not months. We can show it working on your own conversations within days.

PROMISE 5 OF 10

5

“Our engineers have capacity.”

Spare capacity looks free on paper. It comes out of your product roadmap.

AN ILLUSTRATIVE TIMELINE

This quarter

Two engineers are free. Perfect timing.

Year 1

The build takes around 180 engineering weeks.

Year 2

A customer-facing feature slips to next year.

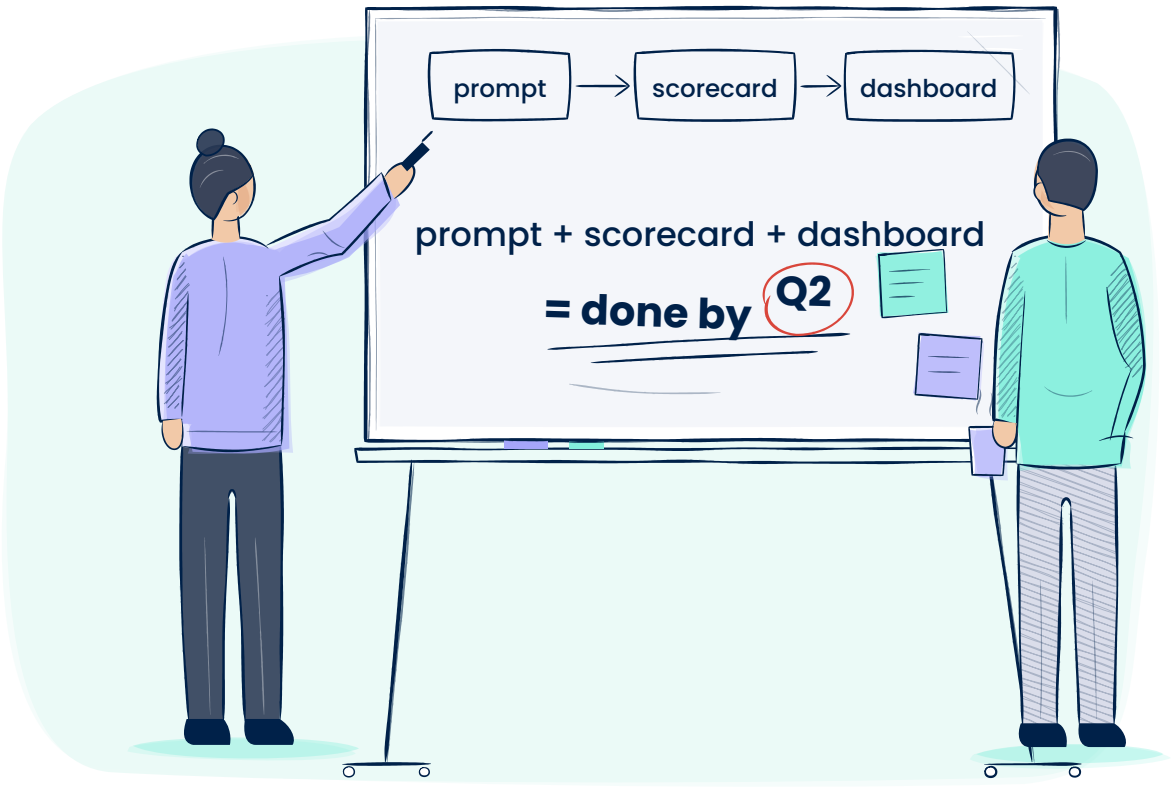


Figure 8 Around 180 engineering weeks in year one, taken from somewhere else on the roadmap.

Spare engineering capacity wins the argument on paper. Two engineers are between projects, the timing looks right, and the build doesn't need a new budget line. The cost is still there. It just shows up somewhere else.

In practice, year one of this build takes roughly 180 engineering weeks off your roadmap. That's time that would otherwise go into the features your own customers are waiting for. And the work doesn't stop at launch. A live quality programme needs fixes, scorecard changes, new integrations and a response to every model change, so the engineering commitment carries on for as long as the tool does.

There's a quieter risk too. Internal builds tend to live in a few people's heads, including the logic behind the prompts and the reasons for each workaround. When those people move teams or leave, your quality programme loses its owner overnight, and whoever comes next inherits a system they didn't design.

None of this is a criticism of the engineers. It's a question of where their time does the most good, for your customers and for your QA team.

> *Around 180 engineering weeks in year one, taken from features your own customers are waiting for.*

Questions to ask your team

1. Which customer-facing projects get dropped to build and run this?
2. Who owns the roadmap and writes the specs for the next five years?
3. What happens when the people who built it move on?
4. Is QA software the best use of our engineers' time?

THE PROOF POINT

Your engineers keep shipping your product. Your QA team spends its time improving the customer experience, not maintaining software.

PROMISE 6 OF 10

6

“Security’s covered.”

QA touches every customer conversation, so its security bar is among the highest in your stack.

AN ILLUSTRATIVE TIMELINE

The prototype

It runs on a test dataset. No real customers yet.

Procurement

The security questionnaire arrives.

The gap

SSO, permissions, audit logs, pen testing. All still to build.



Figure 9 The conversations QA reads carry personal details, payments, complaints and, sometimes, a vulnerable customer.

A prototype usually runs on a test dataset. There are no real customers in it, no procurement team and no auditors. Security looks covered because nothing has tested it yet.

Production is different. QA processes every customer conversation, including personal details, payment conversations, complaints and vulnerable customers. That puts its security bar among the highest of any system you run, and the security questionnaire that arrives during procurement will reflect it.

The system needs encryption, role-based access, audit logs, data residency and independent testing from day one, not on the roadmap. Single sign-on, multi-factor authentication and permissions all have to be built, tested and maintained. Permissions in particular have to reflect who should see which conversations, which gets complicated quickly in a large operation.

Then someone has to evidence all of it, every year. Certifications and penetration tests aren't one-off pieces of work. They recur, and they need an owner who understands both the system and the standard it's being held to. In a build, that owner usually doesn't exist yet.

> *A prototype has no real customers, no procurement team and no auditors. Security looks covered because nothing has tested it yet. The first security questionnaire is where that changes.*

Questions to ask your team

1. Will it pass our own security review?
2. Who maintains SSO, permissions and audit logs?
3. Where is conversation data stored, and who can see it?
4. Who evidences all of this to auditors each year?

THE PROOF POINT

evaluagent holds ISO/IEC 27001:2022, SOC 2 Type II and Cyber Essentials Plus, with data residency in the US, EU or Australia, AES-256 encryption at rest, TLS 1.2+ in transit and annual CREST-certified penetration testing. Details at trust.evaluagent.com.

PROMISE 7 OF 10

7

“Agents will trust the scores.”

A score nobody can explain soon becomes an employee relations problem.

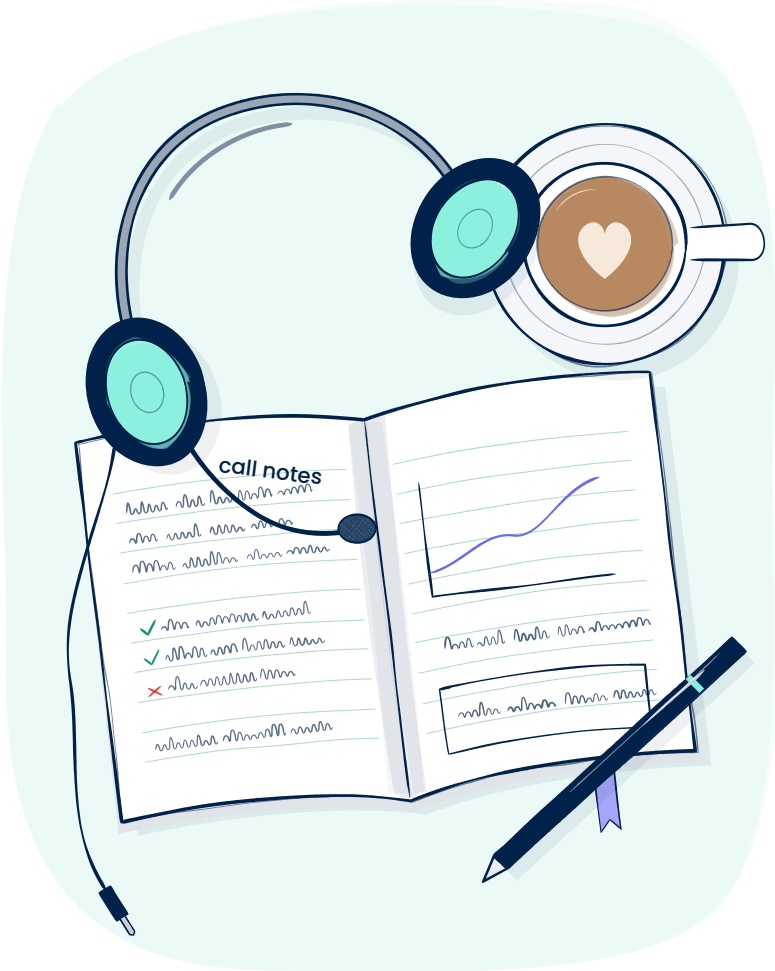


Figure 10 An agent asks why they failed a call. The answer needs evidence.

AN ILLUSTRATIVE TIMELINE

Go-live
Scores go live for the whole team.

Week 3
An agent asks why they failed a call.

Week 4
Nobody can show the evidence. HR gets involved.

> *When a score affects someone’s pay or their record, they’ll want to know how it was reached.*

QA scores drive coaching, performance reviews, incentives and sometimes disciplinary decisions. When a score affects someone's pay or their record, they'll want to know how it was reached. An inaccurate or unexplainable score becomes an employee relations risk the moment it's challenged.

So agents, team leads and compliance all need to see why a score was given and the evidence behind it. They need to be able to challenge it, and someone needs to be able to override it, with the override recorded. Disputes need a process and an owner. Calibration needs a regular rhythm, so analysts and the AI stay aligned over time.

None of that appears in a scoring prototype, yet all of it decides whether people use the scores at all. If agents conclude the scores are arbitrary, team leads stop coaching from them and the programme drifts back to spreadsheets.

Then there's training. Every QA analyst, team lead and agent needs to understand what the scores mean and how to work with them. New starters arrive every month and processes change, so the training never finishes. Someone has to own it.

> *"Feedback only lands if agents believe the score behind it. Scores people can't question give good agents one more reason to leave.*

WHY IT MATTERS

Trust in the score is an employee relations question

80%

of employees who say they've had meaningful feedback in the past week are fully engaged.

Gallup, 2023

39%

average contact centre agent attrition in 2024, reported by 400 contact centre leaders.

NiCE, 2025

High-risk

is how the EU AI Act classes AI that monitors and evaluates workers' performance. Employers must tell staff and their representatives before using it.

EU AI Act, Annex III 4(b), Article 26(7)

Questions to ask your team

1. Can an agent see why they got that score?
2. How do disputes work, and who resolves them?
3. Can a person override the AI, and is that recorded?
4. Who trains QA analysts, team leads and agents, and keeps doing it?

THE PROOF POINT

Explainable scores, dispute and calibration workflows, and human override are in place on day one.

PROMISE 8 OF 10

8

“We can judge our own AI agents.”

As AI agents take on more conversations, someone other than their builders needs to judge them.

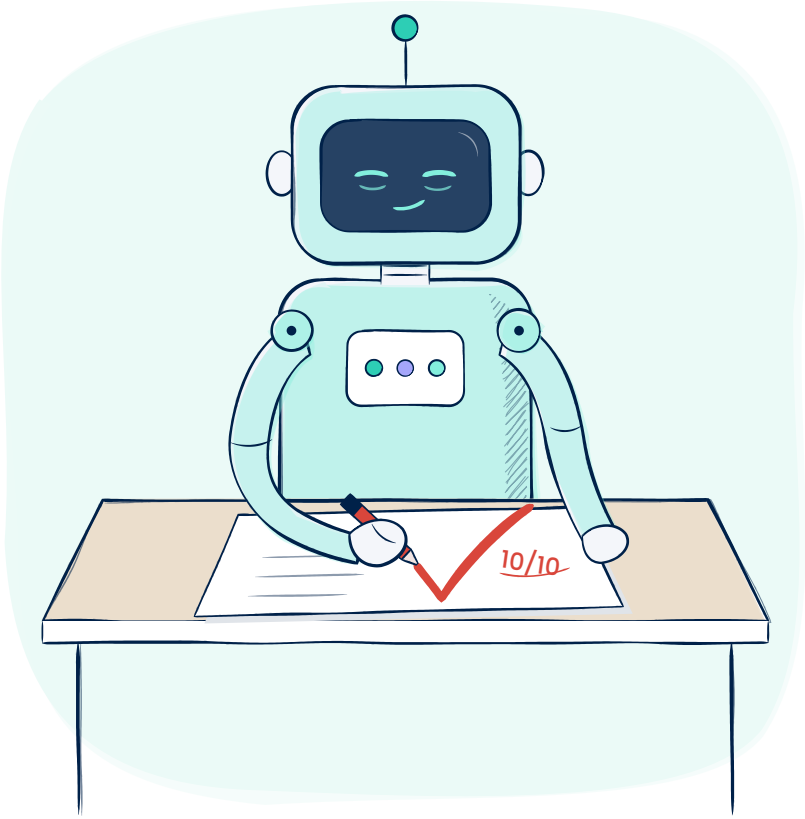


Figure 11 Marking its own homework.

AN ILLUSTRATIVE TIMELINE

Launch
We launch an AI agent for billing queries.

Month 1
We score it with the tool we built alongside it.

Month 2
It gives itself full marks.

> *A QA tool built by the same team, alongside the same agents, ends up marking its own homework.*

As bots and AI agents handle more of your conversations, they need the same quality oversight as your people. Customers don't care whether a person or a bot let them down.

The obvious route is to evaluate AI agents with the tool you're building anyway. The trouble is independence. A QA tool built by the same team, alongside the same agents, ends up marking its own homework. The people who designed the bot also designed the test, and they'll have carried the same assumptions into both.

Independence matters for consistency as well. If human conversations are judged one way and AI conversations another, you can't compare them. You can't tell whether moving a contact type to an AI agent made things better or worse for the customer, which is usually the question the board will ask.

There's an incentive question too. The team that launched the bot wants it to succeed. That's healthy in a product team, but it's the wrong position for the people judging its quality. A quality judge needs to be free to say the bot is failing, and to be believed when it says the bot is fine.

> *A quality judge needs to be free to say the bot is failing, and to be believed when it says the bot is fine.*

Questions to ask your team

1. How will we evaluate our AI agents independently?
2. Will human and AI conversations be judged against the same standard?
3. Who would tell us the bot is failing, and would they want to?

THE PROOF POINT

evaluagent evaluates human and AI agents in one platform, independently of whoever built the bot.

PROMISE 9 OF 10

9

“If others can, we can.”

Some QA vendors have moved their focus away from AutoQA. Doing it well is a full-time job.

AN ILLUSTRATIVE TIMELINE

The pitch

“How hard can it be?”

The market

Vendors juggling other products shift their focus away from AutoQA.

Our build

A side project inherits the same hard problems.



Figure 12 Year three. The builder has moved on, and the tool is still running.

"How hard can it be?" is a fair question to ask about a prototype. It's a harder one to answer about a product that has to keep working for years.

Some QA software vendors have recently moved their focus away from AutoQA. Most were building other products at the same time, and keeping AutoQA accurate, explainable and compliant on top of those newer priorities proved hard to sustain. These are companies with product teams, engineering budgets and customers paying them to get it right.

A side project inside a contact centre runs into the same problems, with fewer people to solve them and a day job competing for their time.

The hard problems don't show up in the first version. They sit in accuracy and drift, governance, dispute handling and adoption, and they arrive gradually once the tool is in daily use. These are the areas where decades of contact centre quality experience earn their keep, because they depend on knowing how QA teams, team leads and agents work day to day.

Then there's year three. A build that started with real enthusiasm can become a project nobody owns. It keeps running and keeps scoring, with no one improving it and no one quite sure who to ask when something looks wrong.

> "
>
> *Keeping AutoQA good is a full-time job. As a side project, it meets the same hard problems with fewer people.*

Questions to ask your team

1. If vendors with whole product teams found AutoQA hard to sustain alongside other work, what makes our plan different?
2. Who will keep improving it in year three?
3. What happens if it becomes a project nobody owns?

THE PROOF POINT

Quality management is our whole business. While others step back from AutoQA, we're investing in it every day.

PROMISE 10 OF 10

10

“We’ll know if it breaks.”

AI fails silently. QA is meant to be the thing that notices.

AN ILLUSTRATIVE TIMELINE

Monday
The dashboard is all green.

Wednesday
A complaint goes public on social media.

Thursday
QA missed the pattern for weeks.

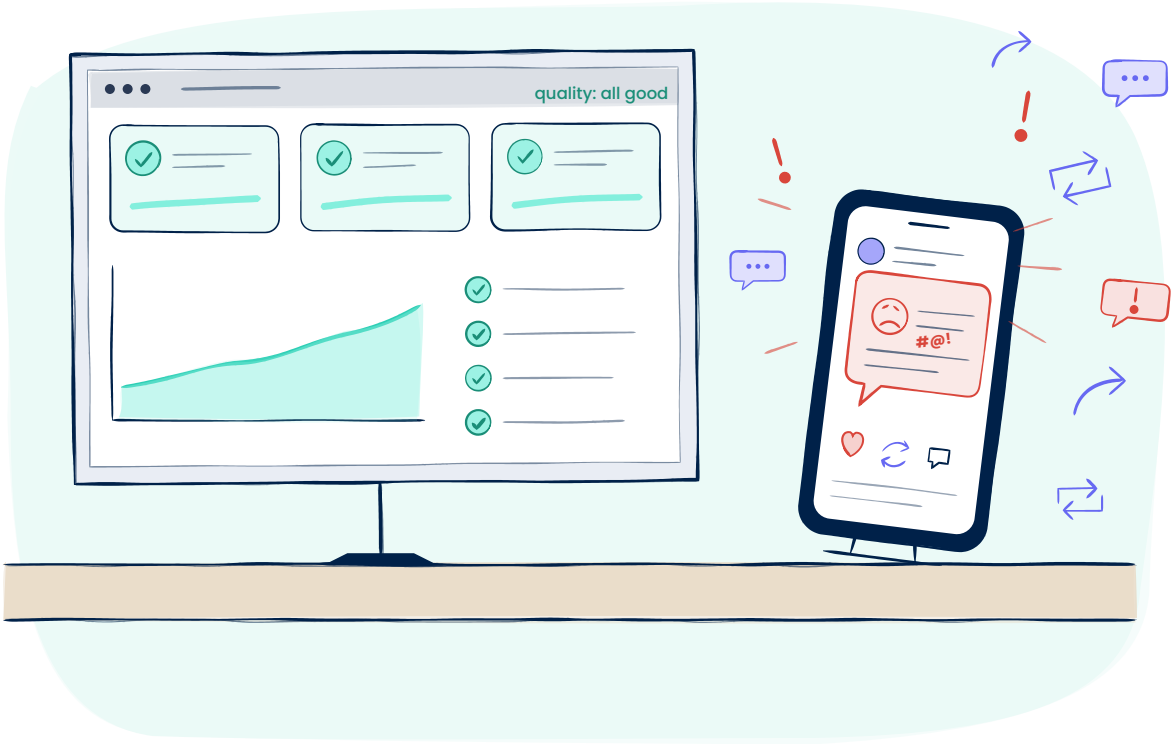


Figure 13 A green dashboard on Monday. A public complaint on Wednesday.

AI fails silently. Model updates, prompt changes and drift can move scores with no error message. The dashboard stays green while the scoring underneath has changed.

That matters more for QA than for most systems, because QA is your early-warning system. It's how you spot a vulnerable customer being handled badly, a product being mis-sold or a compliance breach starting to repeat. When QA misses those things, it fails in front of your customers, your board and your regulator, often weeks after the problem began.

The consequences can be large. In 2024 the FCA fined a UK motor finance firm £5.4m, plus £21.5m in redress to around 110,000 customers, for failing customers in financial difficulty. Your QA record is the evidence of how you look after customers, and you may one day need to show how each score was reached, who reviewed it and what changed.

So the build needs continuous accuracy monitoring rather than a check at launch, and an audit trail that holds up under scrutiny. It also needs someone with a reason to raise the alarm. If the team that built the tool is the only one watching it, ask whether they'd want to be the ones to say it's failing.

> *Your QA record is the evidence of how you look after customers.*

Questions to ask your team

1. If it started failing our customers tomorrow, who would tell us?
2. Would they have any incentive to?
3. How do we monitor accuracy continuously, not just at launch?
4. Can we show a regulator how each score was reached, who reviewed it and what changed?

THE PROOF POINT

Accuracy monitoring, drift reporting and a full audit trail, so problems show up in your data before your customers tell you.

THE CHECKLIST

Ten questions: is the build case *ready*?

Take this page into the meeting. For each question, decide whether the build case has a confident, costed answer, or not yet.

	QUESTION	CONFIDENT, COSTED ANSWER	NOT YET
1	**ACCURACY** How will we know it's accurate, and who checks it?	☐	☐
2	**MODEL CHANGE** Who re-validates every scorecard when the model changes?	☐	☐
3	**TOTAL COST** Have we costed engineering, inference, security, reporting and support beyond year one?	☐	☐
4	**TIME TO VALUE** When will we be live on 100% of conversations?	☐	☐
5	**OPPORTUNITY COST** Which customer-facing projects get dropped to build this?	☐	☐
6	**SECURITY** Will it pass our own security review?	☐	☐
7	**ADOPTION** Can agents see why they got a score, and challenge it?	☐	☐
8	**INDEPENDENCE** How will we evaluate our AI agents without marking our own homework?	☐	☐
9	**OWNERSHIP** Who owns the roadmap for the next five years?	☐	☐
10	**EARLY WARNING** If it started failing our customers tomorrow, who would tell us?	☐	☐

READING THE RESULT

0–2 "not yet"

A well-tested plan. Bring it to us and we'll model both routes side by side.

3–6 "not yet"

There's real air in this plan. Pressure-test those promises first.

7–10 "not yet"

Mostly air. Burst the bubble before the business case reaches the board.

NEXT STEP

Bring us your *build estimate*

If you're weighing a build, bring us your requirements and your build estimate. We'll work through both routes with you, using your numbers rather than ours.



WHAT A SESSION INCLUDES

- ✓ **A side-by-side cost model.** Both routes, built from your own figures.
- ✓ **Your requirements mapped against the platform.** What's covered today, and where the gaps are.
- ✓ **Your own conversations scored.** So you can judge accuracy for yourself.

If the build case holds up, you'll have tested it properly. If it doesn't, you'll find out before it reaches the board.

BOOK A SESSION

evaluagent.com/build-vs-buy →

evaluagent

SOURCES AND ASSUMPTIONS

Costs. Cost figures are an illustrative example for a 300-seat contact centre handling around 80,000 conversations a month. They assume £95k fully loaded per engineer, £110k per AI specialist, and LLM cost of around £0.04 per conversation, rising with prompt complexity. The evaluagent figure assumes an illustrative subscription of £100k a year and a 0.2 FTE programme owner at £12k a year. Both routes exclude transcription, which either would need. The build figure doesn't cost re-validation after model changes. Your numbers will differ.

Timelines. The build timeline in chapter 4 is an estimate, not a benchmark. It assumes the team costed in chapter 3, three engineers and an AI/ML specialist, working through the stages in Figure 6, some in parallel. Four people over roughly 45 working weeks gives the figure of around 180 engineering weeks in year one. The evaluagent figure reflects a standard implementation using existing integrations. The short timelines that open each chapter are illustrative composites, not case studies.

Employee relations (chapter 7). Gallup, "A Great Manager's Most Important Habit", 2023. NiCE, "Managing the Modern Contact Center: Current Employer Trends", 2025, a survey of 400 contact centre leaders. EU AI Act, Annex III point 4(b) and Article 26(7).

Regulation (chapter 10). FCA figures come from the FCA's press release of October 2024.

Security. Certifications and controls are listed at trust.evaluagent.com.