

Trustworthy GenAI: A Quality Engineering Framework for Enterprise AI Products

Harshit Dubey

Department of Engineering

University of Petroleum & Energy Studies

Abstract

Generative AI has moved rapidly from experimentation into enterprise products. Organizations are using Large Language Models (LLMs) for employee assistants, customer support, knowledge search, software engineering, document processing, content generation, analytics, and decision support.

However, the methods traditionally used to assure software quality were not designed for systems that generate probabilistic outputs.

A conventional software application can often be validated against deterministic expected results : given input X , the application should produce output Y . A GenAI application may produce several acceptable responses to the same input. It may also produce a fluent, confident and completely incorrect response. Its behavior can change when the underlying model, prompt, retrieval data, system instructions, safety configuration or surrounding application changes.

This changes the fundamental question for Quality Engineering.

The question is no longer simply:

"Does the software work?"

It becomes:

"Can we demonstrate that the AI product is sufficiently accurate, reliable, safe, secure, useful and controllable for its intended business context?"

This paper proposes a practical **Trustworthy GenAI Quality Engineering Framework** that integrates traditional software quality practices with AI-specific evaluation, risk management, security testing, human evaluation, production monitoring and business validation.

The framework is organized around seven quality dimensions:

1. **Truthfulness & Accuracy**
2. **Reliability & Robustness**
3. **User & Business Value**
4. **Safety & Security**
5. **Transparency & Traceability**
6. **Quality Engineering**
7. **Evaluation & Continuous Monitoring**

The framework does not attempt to replace established AI governance approaches. Instead, it provides an execution-oriented bridge between AI governance and day-to-day product delivery. It aligns naturally with established resources

such as the NIST AI Risk Management Framework and its Generative AI Profile, which emphasize managing AI risks throughout the lifecycle, and ISO/IEC 42001, which provides requirements for an AI management system.

The objective is simple:

Make trust measurable, make quality executable, and make production readiness evidence-based.

1. Introduction

Enterprise adoption of Generative AI has created an unusual situation for technology organizations.

The technology can demonstrate impressive capability very quickly, but capability is not the same as product readiness.

A prototype that generates a convincing answer in a demonstration may fail when exposed to:

- ambiguous questions;
- incomplete information;
- conflicting source documents;
- malicious prompts;
- sensitive information;
- unexpected user behavior;
- new document versions;
- model changes;
- high traffic;
- long conversations;
- unsupported business questions;
- multilingual inputs;
- adversarial users.

This creates a gap between **AI capability** and **AI product quality**.

Traditional Quality Assurance organizations are accustomed to thinking in terms of requirements, test cases, expected results, defects, regression testing and release criteria.

GenAI requires these practices, but they are not sufficient.

For example, consider an enterprise HR assistant.

A traditional test might be:

Input: "What is the annual leave entitlement?"

Expected result: "25 days."

But the quality problem becomes significantly more complicated when the assistant responds:

"Employees receive 25 days of annual leave."

The response may look correct, but additional questions immediately arise:

- Was the answer retrieved from the current HR policy?

- What happens if the policy was updated yesterday?
- What if the employee belongs to another country?
- What if the user asks about a contractor?
- Can the system cite the source?
- What happens when no relevant policy exists?
- Will the model invent an answer?
- Can an employee retrieve another employee's confidential information?
- What happens if the user attempts prompt injection?
- How much confidence should the organization have in the answer?

This is where Quality Engineering must evolve.

2. The Central Problem: GenAI Quality Is Multi-Dimensional

Traditional application testing often focuses on whether a system performs the intended function.

GenAI introduces additional dimensions.

A response can be:

- grammatically excellent but factually wrong;
- factually correct but irrelevant;
- relevant but unsupported by evidence;
- accurate but unsafe;
- safe but unusably vague;
- useful but excessively expensive;
- correct for one prompt but fragile to small variations;
- correct today but incorrect after a model update.

Therefore, a GenAI product should not have a single "quality score."

Quality should instead be viewed as a multidimensional system.

I propose the following model:

The Trustworthy GenAI Quality Equation

Trustworthy GenAI Product =

Functional Quality × AI Quality × Safety × Security × Business Value × Operational Reliability

The multiplicative representation is intentional.

If an enterprise assistant has excellent accuracy but exposes confidential information, it is not trustworthy.

If it is secure but consistently provides incorrect answers, it is not trustworthy.

If it is accurate and safe but has unacceptable latency and cost at production scale, it is not a viable product.

Trust is therefore constrained by the weakest critical dimension.

3. Why Traditional QA Alone Is Not Enough

Traditional QA remains essential.

Regression testing, API testing, integration testing, performance testing, security testing, accessibility testing and usability testing do not disappear because an application uses an LLM.

They become the foundation.

The difference is that GenAI introduces another layer of quality evaluation.

Traditional application

Requirement → Code → Deterministic behavior → Test → Expected result

GenAI application

Business intent → Prompt/model/data/context → Probabilistic behavior → Evaluation → Risk-based acceptance

This distinction is critical.

A GenAI response cannot always be compared with one exact expected string.

Instead, evaluation may need to determine:

- Is the answer factually correct?
- Does it answer the question?
- Is it grounded in the approved source?
- Does it contain unsupported claims?
- Is it safe?
- Is it appropriately cautious?
- Does it follow the system's instructions?
- Does it expose sensitive information?
- Is it consistent across relevant variations?

This means QA teams need to evolve from **test execution** toward **quality engineering and evaluation engineering**.

4. The Trustworthy GenAI Quality Engineering Framework

The proposed framework contains seven dimensions.

4.1 T — Truthfulness & Accuracy

The system should provide responses that are correct for the intended business context.

This includes:

- factual accuracy;
- calculation accuracy;
- retrieval accuracy;

- source alignment;
- groundedness;
- citation correctness;
- appropriate handling of uncertainty;
- hallucination

Recommended metrics

Metric	Example target
Answer correctness	$\geq 95\%$
Grounded response rate	$\geq 95\%$
Citation accuracy	$\geq 98\%$
Unsupported claim rate	$\leq 2\%$
Critical factual error rate	0%

These are **illustrative thresholds**, not universal industry standards. Thresholds should be risk-adjusted for the specific product.

A medical decision-support application should have very different tolerances from a creative-writing assistant.

4.2 R — Reliability & Robustness

A trustworthy GenAI application should not behave correctly only for the exact prompt used by the development team.

The system should remain useful when users:

- rephrase questions;
- make spelling errors;
- use incomplete sentences;
- change the order of information;
- provide contradictory context;

- ask follow-up questions;
- switch languages;
- use long conversations;
- attempt unusual but legitimate workflows.

Example

Consider:

"What is our refund policy?"

A test suite should include variations such as:

"Can I get my money back?"

"How long do I have to cancel?"

"Customer bought product 28 days ago. Can they return it?"

"What if the product is defective?"

The goal is not identical responses.

The goal is **consistent business behavior**.

Recommended metrics

- Response success rate
 - Robustness pass rate
 - Context retention rate
 - Multi-turn task completion rate
 - Regression rate after model/prompt changes
 - Failure recovery rate
-

4.3. U — User & Business Value

Technical quality does not automatically equal product value.

A chatbot can have excellent benchmark scores and still fail to solve the user's problem.

Therefore, evaluation should include:

User metrics

- task completion rate;
- user satisfaction;
- escalation rate;
- abandonment rate;
- repeat-question rate;
- time saved;

- successful self-service rate.

Business metrics

- cost per successful interaction;
- reduction in support workload;
- processing time reduction;
- productivity improvement;
- revenue impact;
- error reduction;
- employee adoption.

This is particularly important for Product Managers.

The product should not be optimized merely for **better AI responses**.

It should be optimized for **better outcomes**.

4.4 S — Safety & Security

Security and safety must be tested explicitly.

GenAI applications create attack surfaces that are different from traditional applications.

Examples include:

- prompt injection;
- indirect prompt injection;
- sensitive information disclosure;
- excessive agency;
- insecure tool use;
- data leakage;
- malicious document content;
- unsafe generated content;
- authorization bypass;
- model manipulation.

Security testing should therefore be integrated into the GenAI testing lifecycle rather than treated as a final-stage activity.

Security quality gates

A production release should not proceed when critical security vulnerabilities remain unresolved.

Examples:

- zero critical data leakage findings;
- zero unresolved critical authorization bypasses;

- zero unresolved critical prompt-injection paths affecting protected operations;
 - sensitive data handling validated;
 - logging and monitoring validated;
 - tool permissions validated.
-

4.5 T — Transparency & Traceability

Enterprise users increasingly need to understand:

- where an answer came from;
- which data was used;
- which model generated the answer;
- which prompt/version was used;
- when the source data was last updated;
- whether the system used a tool;
- what happened when the model failed.

Traceability becomes particularly important during incident investigation.

For example:

A customer reports that the AI gave an incorrect answer.

The organization should ideally be able to reconstruct:

User prompt → System prompt/version → Model/version → Retrieved documents → Tool calls → Model response → Safety checks → Final response

Without this information, root-cause analysis becomes extremely difficult.

4.6. Q — Quality Engineering

Quality should not be a testing phase at the end of the delivery lifecycle.

It should be embedded from product discovery through production.

A practical lifecycle is:

Discover

Identify business value and risk.

↓

Define

Define quality characteristics and acceptance thresholds.

↓

Design

Design prompts, retrieval, guardrails and application behavior.

↓

Evaluate

Create evaluation datasets and baseline measurements.

↓

Test

Perform functional, AI, security, performance and adversarial testing.

↓

Release

Apply risk-based quality gates.

↓

Monitor

Monitor real-world quality.

↓

Learn

Feed production findings back into the product and evaluation suite.

This is **continuous AI Quality Engineering**.

4.7 E — Evaluation & Continuous Monitoring

The most important operational difference between traditional applications and GenAI systems is that testing cannot stop at release.

Model behavior can change because of:

- model upgrades;
- prompt changes;
- retrieval changes;
- embedding changes;
- knowledge-base changes;
- system-instruction changes;
- safety-policy changes;
- external APIs;
- user behavior.

Therefore:

The production system becomes part of the test environment.

Production monitoring should measure both technical and AI behavior.

5. A Practical GenAI Quality Architecture

A mature enterprise implementation can use five quality layers.

Layer 1 — Conventional Software Quality

Includes:

- unit testing;
- API testing;
- integration testing;
- UI testing;
- performance testing;
- accessibility;
- security;
- infrastructure testing.

Layer 2 — AI Evaluation

Includes:

- correctness;
- relevance;
- groundedness;
- hallucination;
- summarization quality;
- instruction following;
- response consistency.

Layer 3 — Adversarial Testing

Includes:

- prompt injection;
- jailbreak attempts;
- data extraction;
- malicious documents;
- context manipulation;
- authorization abuse.

Layer 4 — Human Evaluation

Domain experts evaluate representative outputs.

This is particularly important when automated metrics cannot fully capture business correctness.

Layer 5 — Production Quality Monitoring

Monitor:

- failure rate;
 - user feedback;
 - escalation;
 - hallucination reports;
 - latency;
 - cost;
 - safety events;
 - model drift;
 - knowledge freshness.
-

6. Sample Quality Gates

A major weakness in many AI projects is the absence of explicit release gates.

The statement:

"The model looks good."

is not a quality criterion.

A better approach is to establish measurable gates.

Gate 1 — Business Readiness

Before detailed AI testing begins:

- business objective documented;
- target users defined;
- prohibited use cases defined;
- success metrics defined;
- risk classification completed;
- human escalation path defined.

Release decision: Product Owner + Business Owner.

Gate 2 — Evaluation Dataset Readiness

The team should have a representative evaluation dataset.

Example:

Category	Minimum examples
Normal questions	500

Category	Minimum examples
Edge cases	100
Negative cases	100
Ambiguous questions	100
Safety/security cases	100
Out-of-scope questions	100

The numbers are illustrative. The correct database size depends on risk, usage volume and domain complexity.

More important than volume is **coverage**.

The evaluation dataset should represent real user behavior rather than only ideal prompts written by the development team.

Gate 3 — AI Quality

Example release thresholds:

Metric	Target
Correctness	$\geq 95\%$
Groundedness	$\geq 95\%$
Relevance	$\geq 95\%$
Critical factual errors	0
Citation correctness	$\geq 98\%$
Appropriate refusal	$\geq 95\%$

Critical use cases may require stricter thresholds.

Gate 4 — Safety & Security

Example:

Test	Gate
Critical data leakage	0
Critical authorization bypass	0
High-risk prompt injection	0 unresolved
Unsafe prohibited output	Within approved tolerance
Sensitive information exposure	0 critical cases

Test	Gate
Audit logging	100% required events
Tool authorization	100% validated

Gate 5 — Performance

Example:

Metric	Example threshold
p95 latency	≤ 5 seconds
Error rate	$\leq 1\%$
Timeout rate	$\leq 0.5\%$
Availability	$\geq 99.5\%$
Cost per successful task	Within business budget

Again, thresholds should be tailored to the use case.

7. Enterprise Scenario 1: Internal Knowledge Assistant

Consider an organization deploying an AI assistant that answers questions using internal policies and knowledge articles.

Business goal

Reduce employee time spent searching internal documentation.

Architecture

Employee → AI Assistant → Retrieval Layer → Enterprise Knowledge Base → LLM → Guardrails → Response

Primary risks

- outdated information;
- hallucination;
- confidential information leakage;
- incorrect policy interpretation;
- poor retrieval;
- unauthorized document access.

Quality strategy

The QA team creates evaluation sets covering:

1. straightforward policy questions;
2. questions requiring multiple documents;

3. outdated documents;
4. conflicting documents;
5. questions with no answer;
6. unauthorized information;
7. ambiguous employee questions;
8. prompt-injection attempts.

Example quality gate

The product cannot move to production unless:

- $\geq 95\%$ of benchmark questions are correctly answered;
- $\geq 95\%$ are grounded in approved sources;
- 100% of critical confidential-data scenarios are blocked;
- $\geq 98\%$ of citations point to the correct source;
- unsupported questions trigger an appropriate response;
- access control is validated across user roles.

Practical lesson

One of the most common mistakes is testing only:

"Does the chatbot answer correctly?"

The more important question is:

"Does the chatbot know when it should not answer?"

A trustworthy enterprise assistant needs both capabilities.

8. Enterprise Scenario 2: Customer Service Copilot

Consider an AI copilot used by customer service representatives.

The system summarizes customer conversations and recommends responses.

Risks

A wrong answer may result in:

- customer dissatisfaction;
- financial loss;
- incorrect refunds;
- regulatory exposure;
- reputational damage.

Quality dimensions

The QA strategy should therefore include:

Summarization quality

Does the summary preserve critical information?

Recommendation quality

Does the recommended response follow company policy?

Context quality

Does the response use the correct customer history?

Safety

Does it avoid making unauthorized commitments?

Human control

Can the agent review and override the recommendation?

Critical quality principle

The system should not be evaluated only on:

"How good is the generated answer?"

It should also be evaluated on:

"How effectively does the system support the human decision-maker?"

This distinction is essential for enterprise AI.

9. Building a GenAI Evaluation Dataset

One of the strongest predictors of GenAI testing maturity is the quality of the evaluation dataset.

A useful dataset contains:

Positive cases

Questions the system should answer.

Negative cases

Questions the system should reject or redirect.

Boundary cases

Questions near the limits of the supported domain.

Adversarial cases

Prompts designed to manipulate the system.

Ambiguous cases

Questions where clarification is preferable to guessing.

Freshness cases

Questions affected by recently changed information.

Multi-turn cases

Conversations where previous context matters.

Role-based cases

Different users with different authorization levels.

The evaluation dataset should become a **living product asset**, not a one-time QA artifact.

Every production incident should be considered for inclusion in the regression suite.

10. Human Evaluation

Automated evaluation is valuable, but it should not become a substitute for expert judgment.

Human evaluators should be used when:

- domain expertise is required;
- context is complex;
- safety implications are significant;
- subjective quality matters;
- automated evaluators have known limitations.

A practical evaluation rubric can score each response from 1–5:

Dimension	1	3	5
Correctness	Incorrect	Partially correct	Fully correct
Relevance	Irrelevant	Partially relevant	Directly relevant
Grounding	Unsupported	Partially supported	Fully supported
Clarity	Confusing	Acceptable	Clear
Safety	Unsafe	Minor concern	Safe
Completeness	Major gaps	Some gaps	Complete

Dimension	1	3	5
Correctness	Incorrect	Partially correct	Fully correct
Relevance	Irrelevant	Partially relevant	Directly relevant
Grounding	Unsupported	Partially supported	Fully supported
Clarity	Confusing	Acceptable	Clear
Safety	Unsafe	Minor concern	Safe
Completeness	Major gaps	Some gaps	Complete

The objective is not to eliminate human evaluation.

It is to use humans where human judgment creates the most value.

11. RACI: Enterprise GenAI Quality Responsibilities

One of the most common delivery problems is unclear ownership.

Everyone assumes somebody else owns AI quality.

A practical RACI model is therefore essential.

Roles

- **PO** — Product Owner
- **QA** — QA / Quality Engineering Lead
- **AI** — AI/ML Engineer or Data Scientist
- **DEV** — Application Development Lead
- **SEC** — Security Lead
- **DATA** — Data/Knowledge Owner
- **OPS** — Platform/SRE/Operations
- **GOV** — AI Governance/Risk/Compliance
- **BUS** — Business/Domain SME

RACI Matrix

Activity	PO	QA	AI	DEV	SEC	DATA	OPS	GOV	BUS
Define business objective	A	C	C	C	I	C	I	C	R

Activity	PO	QA	AI	DEV	SEC	DATA	OPS	GOV	BUS
Define AI quality criteria	A	R	C	C	C	C	I	C	C
Risk assessment	A	R	C	C	R	C	C	A/R	C
Evaluation dataset	C	R	R	C	C	R	I	C	R
Functional testing	C	A/R	C	R	C	I	C	I	C
AI evaluation	C	A/R	R	C	C	C	I	C	R
Security testing	I	C	C	C	A/R	C	C	C	I
Data validation	I	C	C	C	C	A/R	I	C	R
Performance testing	C	A/R	C	R	C	I	R	I	I
Release recommendation	A	R	C	C	C	C	C	C	C
Production monitoring	C	R	R	R	C	C	A/R	C	C
Incident management	A	R	C	R	R	C	R	C	C

R = Responsible

A = Accountable

C = Consulted

I = Informed

The exact RACI should be adapted to organizational structure.

The important principle is:

QA should own quality engineering but should not be the only function accountable for product quality.

Quality is a cross-functional responsibility.

12. Conclusion

GenAI quality goes beyond traditional testing and requires a multidimensional quality approach.

Organizations must evaluate accuracy, robustness, safety, security, transparency, and business value.

Traditional QA should evolve into continuous **AI Quality Engineering**.

Quality must be measurable, risk-based, and continuously monitored in production.

Cross-functional ownership is essential to building trustworthy AI products.

The goal is not just capable AI, but **safe, reliable, explainable, and business-ready AI**.

Trustworthy GenAI Quality Engineering is the foundation for responsible enterprise AI adoption.