Applied AI: use it where it helps, and verify what it produces

21 October 2026 · 6 min read · blogs.iictp.org/applied-ai-verify-what-the-machine-produces

The third TPBOK domain, Applied AI & Emerging Tech Integration, is not about knowing the newest model. It is about a durable skill: deciding where a probabilistic tool belongs in a system that must behave predictably, and how to check its output before anyone relies on it.

TL;DR The short version

- Applied AI is about deciding where a probabilistic tool belongs in a system that must behave predictably.
- Start with uses where a mistake is cheap, visible and easy to check, and hold generated code to the same tests and review as human code.
- Treat a model inside a product as an unreliable component, and use published risk frameworks rather than inventing your own.

Read online, listen to it, or change the text size: https://blogs.iictp.org/applied-ai-verify-what-the-machine-produces

The core problem: output is probable, not guaranteed

A traditional function returns the same answer for the same input. A language model returns a likely answer. That is useful, and it means the cost of verification moves to the user. Brooks's point that the hard part of software is its essential complexity applies here too: AI can write code quickly, but someone still has to decide whether the code is right.

Where to start: low-risk, reversible, checkable

A practitioner's first question is not "can AI do this?" but "what happens if it is wrong, and how will we know?" Good first uses have three properties: a mistake is cheap, a mistake is visible, and a person or a test can check the result.

- Writing tests, documentation or boilerplate that a reviewer will read.
- Searching and summarising internal material, with sources shown.
- Suggesting changes in a pull request that still needs human approval.

Checks that belong before anything merges

- Automated tests that fail when behaviour changes, so generated code is held to the same bar as human code.
- Human review of anything touching security, money or personal data.
- Secrets and personal data kept out of prompts and out of generated code.
- A record of which output came from a tool, so problems can be traced.

Designing systems that include a model

When a product calls a model, treat it as an unreliable component. Retrieval-augmented generation, introduced by Patrick Lewis and colleagues in 2020, grounds answers in documents you control. It helps, but it adds its own parts that can fail: how documents are split, how they are ranked, and whether the answer is faithful to the source. Test each part. Measure answers against a fixed set of questions so you can see whether a change made things better or worse.

- **Cost and rate limits.** Put limits and fallbacks around every call.
- **Data privacy.** Decide what may leave your boundary before you integrate.
- **Observability.** Log inputs, outputs and decisions so you can investigate an odd answer.

> Anti-patterns: merging generated code without review; chaining prompts with no validation between steps; dumping every document into a vector store with no structure and hoping retrieval sorts it out.

Follow the published frameworks

You do not have to invent your own risk language. The NIST AI Risk Management Framework and the OWASP list of risks for applications built on language models are public, maintained, and a good starting vocabulary. For evidence on how AI changes delivery performance, read the latest DORA reports directly rather than relying on summaries.

Applied AI & Emerging Tech Integration is 16% of the ATP exam and 18% of the CTP exam, and the largest share (36%) of the specialist AI Practitioner (AIP) exam. See exam facts.

Sources and further reading

1. Brooks, F. P. (1987). “No Silver Bullet: Essence and Accidents of Software Engineering.” IEEE Computer 20(4).
2. Lewis, P. et al. (2020). “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” Advances in Neural Information Processing Systems 33.
3. National Institute of Standards and Technology (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1.
4. OWASP Foundation. OWASP Top 10 for Large Language Model Applications.
5. DORA (Google Cloud). Accelerate State of DevOps reports, published annually.

Published by IICTP, an independent technical credential body, on 21 October 2026. The current version of this post is at https://blogs.iictp.org/applied-ai-verify-what-the-machine-produces. More guides: blogs.iictp.org.