APPLIED AI & PRODUCT SYSTEMS

From To Do Overload to Operational Control

How utilities can bring Oracle Energy and Water exception backlogs under control without giving up control of their data

WHITE PAPER | OCTOBER 2026

RB Fonseca
CONSULTING

Executive summary

Every Oracle Energy and Water implementation eventually meets the same quiet problem. When something goes wrong, the system creates a To Do: a bill fails to generate, a meter reading is missing, a validation rule fires, an inbound message cannot be mapped. The nightly runs create them faster than the team can clear them. Old entries pile up, real problems hide among the noise, and bills slip.

Artificial intelligence looks like the obvious answer, and it is part of one. But utilities are right to be careful. They will not hand customer and operational data to a service outside their control. Cloud customers cannot reach the database at all. And "let the AI fix it" is the wrong goal, because in these systems the usual fixes (retry, suppress, close) can quietly discard data.

We built a working lab to find out what actually helps. The lab is a complete Oracle Energy and Water Customer to Meter (C2M) 25.4 environment loaded with Oracle's DEMO database: a safe place to develop and prove the approach, with real product behavior and no customer data. We tested a ladder of five steps, from no AI at all to guarded automation, with prevention running underneath. Three things stood out:

- **Most of the first win needs no AI.** In our lab, the product's own retry and monitor features cleared hundreds of To Dos once they were switched on, and a triage step showed that almost all of the live backlog sat behind a setting that stopped those features from ever running.
- **The fixes have side effects that must be agreed first.** In our tests, a retry discarded records instead of repairing them, and suppressing a To Do discarded the data that caused it.
- **A language model can prepare the case, and rules can keep it safe.** (A *case* here is our own review file for one To Do: its numbered facts, the model's proposal and the reviewer's decision. It is not a case object in the Oracle products.) Given numbered facts and strict guardrails, a model proposed useful next steps and spotted inconsistencies a person had missed. It was never allowed to change anything. In the lab we also showed a person's approval carrying one narrow action into the product.

This paper summarizes the approach and what we learned. The numbers come from a lab, not from a utility's production system, and are marked as such. A full technical evidence report follows for readers who want the detail.

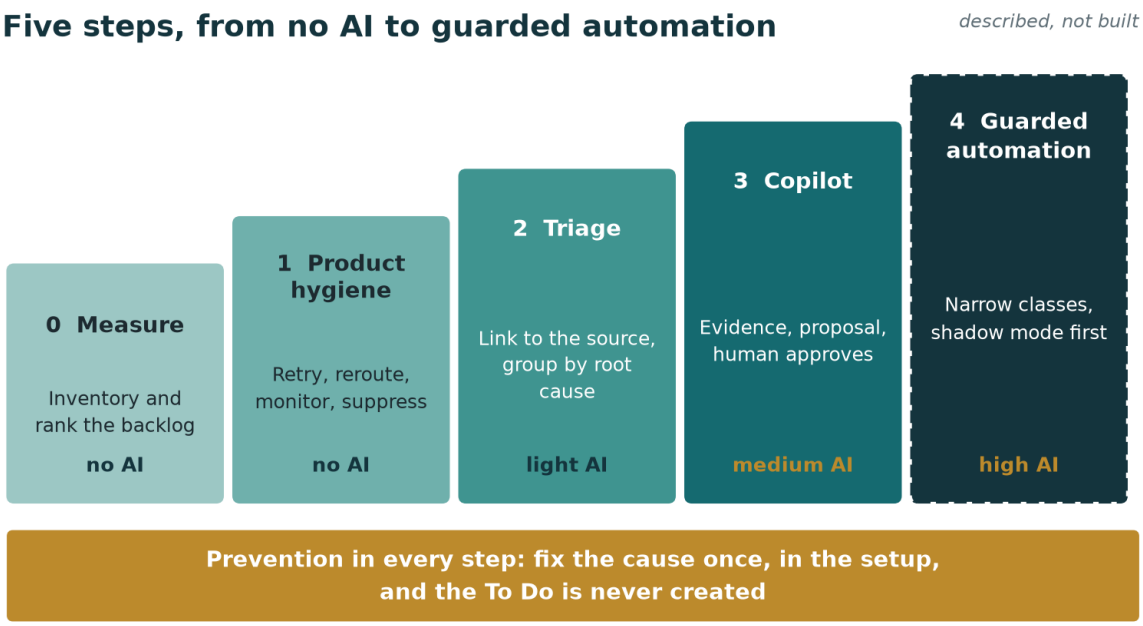


Figure 1. Five steps, from no AI to guarded automation, with prevention running through all of them.

1. The problem: a To Do is a symptom

A To Do is a work item addressed to a role. Oracle Energy and Water creates one when a background process or a status change finds something a person must look at. In a typical implementation they come from exception tables, lifecycle plug-ins, integration errors and users.

Three things make the queue hard to control:

- **Volume.** One bad setting can create thousands of identical entries.
- **Mixed causes.** A high reading, a missing tariff map and a rejected message look alike in a queue but need different people.
- **No owner for the setting.** The team that works the queue rarely owns the configuration that fills it.

Looked at closely, a To Do is rarely the problem itself. It is where the problem surfaces. A billing error often points at a usage transaction, which depends on a validation exception, which depends on a measurement that was rejected at intake. Clearing the end of that chain clears everything that depends on it.

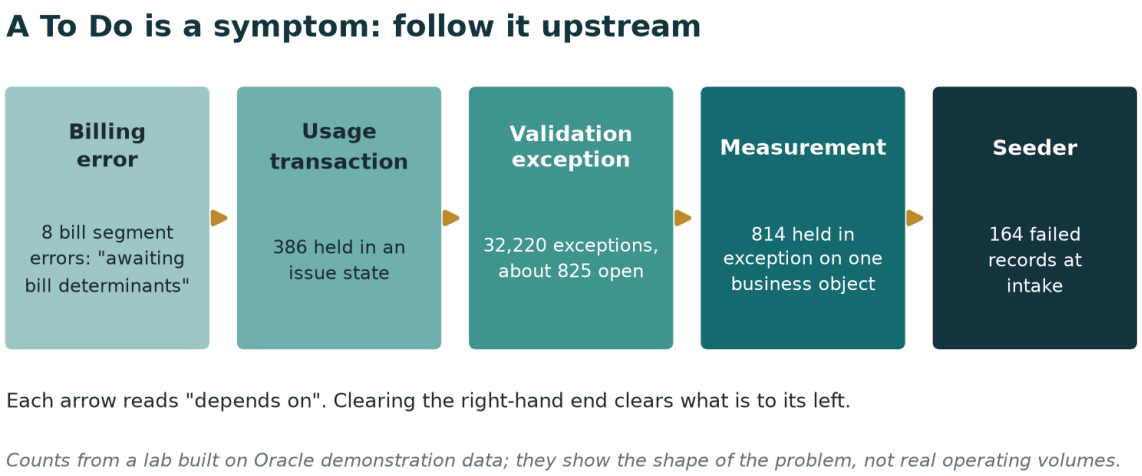


Figure 2. A To Do is a symptom: follow it upstream. Counts from a lab built on Oracle demonstration data.

2. What we built and tested

The lab is a full C2M 25.4 environment loaded with Oracle's demonstration data. It holds about 2,500 To Dos, and the shape of the backlog is familiar: most entries are old, most sit on the meter-data side, and a handful of messages account for most of the volume. We used it to test five steps:

Step	What it does	AI involved
0. Measure	Count and rank the backlog; find the messages that dominate it	None
1. Product hygiene	Use the product's own levers: monitors, retry, rerouting, suppression	None
2. Triage	Link each To Do to its source record, decide if it is still real, group by root cause	Light
3. Copilot	Assemble the evidence, propose a diagnosis and next steps, a person approves	Medium
4. Guarded automation	Carry out narrow, well-understood classes without a person, in shadow mode first	High (described, not built)

Prevention runs through every step. The cheapest To Do is the one that is never created, so each step also looks for the setting that should be fixed once.

Each step stands alone. A utility can stop at any rung and still keep the value.

3. What the lab taught us

A retry that could never run

The product has a standard pattern: a record enters an error state, a To Do is created, a monitor retries the record after a delay, and the To Do closes when the record recovers. In our lab, one run of the monitor batch cleared 164 failed records and closed their To Dos with nobody involved.

But the monitor only runs if a scheduled job is assigned to the state. When our triage linked every open To Do to its source, it found that **98% of the live entries sat on states with no monitor job at all.** The retry was plugged in. Nothing ever ran it. Switching on one job per business object was the largest single lever by volume that the triage found, once the end state of a retry is agreed.

A fix that discards

The fixes carry risks that are easy to miss. A retry on usage transactions did not repair 58 of them: it discarded them. Suppressing a To Do for one message did the same, because the system discarded the record that caused it, while rerouting the same message to another team kept the record. And a To Do monitor that closes entries by age closed 689 To Dos because they were old, not because anything was fixed.

None of this is a flaw in the product. It means every lever needs a written end state ("what happens to the data?") and a business sign-off before it is switched on.

The setting nobody owned

An empty section of a master configuration sent every measurement older than the current date to an error state and created To Dos in a default role that nobody works. In the opposite direction, a reading roughly nine to fifteen times a meter's normal monthly use was accepted with no exception and no To Do at all: only the basic checks ran, and the high and low rule did not. One kind of gap creates noise. The other creates silence. Both are configuration, and both are findable.

4. Privacy and control are part of the design

The most common objection to AI in this setting is the right one: *where does our data go?* Our answer is a design rule, not a promise: **the AI goes to the data, not the data to the AI.**

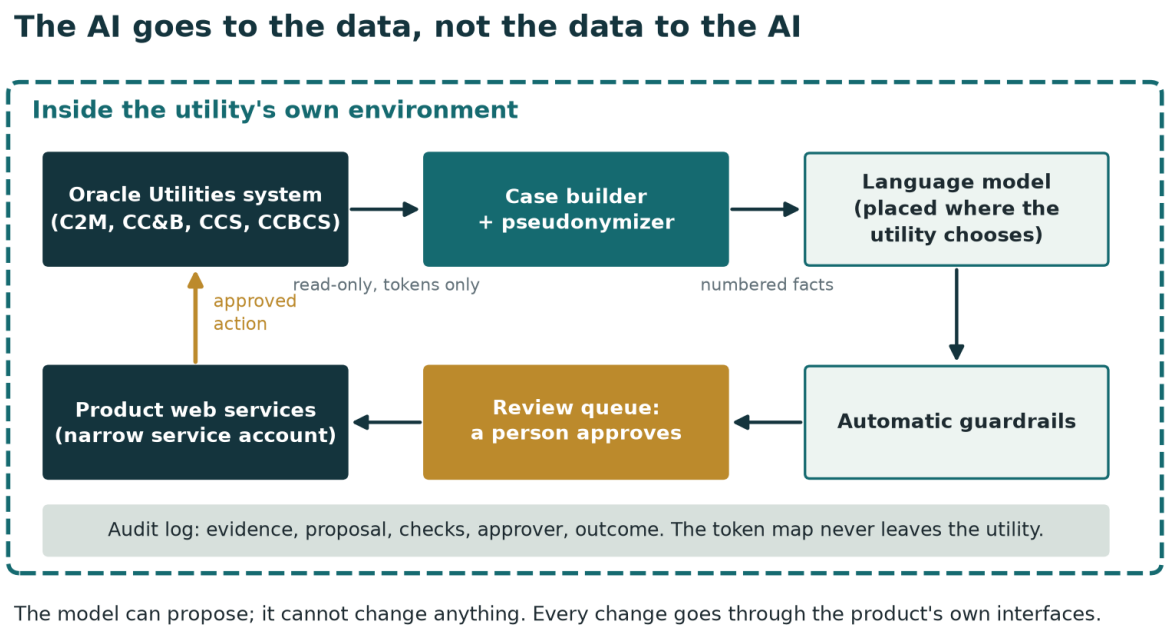


Figure 3. The AI goes to the data, not the data to the AI.

- **Read-only.** Every step except an approved action only reads. The case builder never writes.
- **Pseudonymized.** Identifiers become tokens before anything reaches a model. The mapping stays with the utility.
- **Placed where the utility chooses.** The model can run inside the utility's own tenancy or network, or through a managed model service under its own contract.
- **Cloud-ready.** For software-as-a-service customers who cannot reach the database, Oracle provides sanctioned routes: replication into a customer-owned database, scheduled file export to a customer bucket, and REST services under the customer's own credentials.
- **A person approves.** The model proposes. Approved actions go through the product's own interfaces under a narrow service account, and every step is audited.

5. How this helps client teams

Team	What they get
Operations and billing	A short, ranked list of root causes instead of a long queue; prepared case files for the entries that need judgment
Configuration owners	A list of the settings behind the noise, each with the effect expected from fixing it and the risk
Architects and integrators	A map of how errors flow upstream, and which monitors and routes are missing
Security and privacy	A read-only design with pseudonymized data and an audit trail
Leadership	Measures from the first week: open To Dos by age, share resolved by the product's own features, share of proposals approved

What a prepared case looks like

A *case* in this paper is our own review file for one To Do, not a case object in the Oracle products. For each To Do the copilot collects numbered facts into it: the message with its parameters, the source record and its state, the exceptions, how many other To Dos share the message, whether a monitor is assigned, whether the entry sits in the role it should. The model returns a diagnosis, what is unknown, actions from a fixed list with their end states, and a ready-to-paste log note.

In our six sample cases, the final version passed every automatic guardrail in all 18 answers across three repeats. It caught things a person who knew the lab had missed: a To Do dated before the record it pointed to, and a message whose type did not match the exception on its source. Where the facts did not support confidence, it said so.

From approval to action

We also closed the loop for three narrow actions: adding a log note, forwarding a To Do to another role and back, and completing a stale To Do. A reviewer approved each one, and a small gateway carried it out through the product's own web services, under a service user that can do only that. A separate check confirmed every change, and a full test that also tried things that must be refused found that the product's settings alone did not enforce everything we expected on this path: a broad interface let through an action it should not, and a completion request was not held to the To Do's role. We closed both with narrower interfaces and added checks, and re-tested, which is the point of testing the path before trusting it.

---

6. What could be next

This is a lab result, not a production deployment. Natural next steps are:

- A read-only assessment at a willing utility, using the first two steps.
- The approval path on a real system: log notes, forwarding and completing work in the lab; a production or software-as-a-service tenant, real volumes and the client's own roles are next.
- A pilot that scores proposals against real reviewer decisions on a larger set.
- Configuration changes tested on a non-production copy of a real system.

Each step should be introduced with governance that fits the client environment. The aim is not to replace expert accountability. It is to make expert work faster, clearer and easier to repeat.

The same privacy-first approach underlies the Oracle ERD Builder, our workbench for exploring Oracle Energy and Water data models.

---

7. How to get started

The best starting point is a **read-only assessment**. It needs a bounded scope, a non-production or representative environment, and a small group of operations and configuration owners. An initial conversation can settle:

1. The systems and To Do types in scope.
2. How we can read the data (replica, export, or REST) and who approves access.
3. The outputs you want: a ranked backlog, a configuration findings list, a triage report, or a pilot copilot.
4. Security and governance expectations.

Discuss an assessment with us.

For the numbers behind every claim, read the technical evidence report.

---

8. Need a custom AI solution for your business or technical team?

The same applied-AI approach can be tailored to other workflows: turning scattered system knowledge into focused, reviewable tools that fit how your teams already work. If you have a high-value process slowed by fragmented knowledge, complex configuration or repetitive investigation, RB Fonseca Consulting can help assess and design a practical solution.

Talk with us about a custom AI solution.
