Microsoft Dynamics 365

Designing Evaluation Sets, Metrics, and the Evaluation Blueprint

Agent Evaluation Series - April 29, 2026

Presenters:
Amira Beldjilali
Ahmet Ziya Yıldırım

Moderators: Lutz Eickenberg

Agenda

- Why eval design matters
- A real example: the D365 CE User Onboarding Agent
- Principles: scenario-based sets, coverage balance, synthetic vs real data
- Live demo: AI Workbench Analysis tab
- Amira: the Evaluation Design Document (EDD)
- Q&A and wrap-up

Learning objectives

By the end of this session, you will be able to:

- Design a scenario-based evaluation set that reflects real business processes
- Balance representative and edge-case coverage in your eval set
- Choose between synthetic and real-world derived data and know when each fits
- Fill out an Evaluation Design Document (EDD) as your evaluation blueprint
- Understand how EDD results inform go/no-go decisions

The cost of NOT evaluating

- Agents change the game. FDD/TDD/UAT alone do not catch non-deterministic failures.
- Wrong tool, wrong intent e.g., agent calls Phase2Trigger on a disabled user
- Hallucinated data agent invents a license assignment that does not exist
- Silent drift a model update changes behavior, nobody notices until a customer complains
- Eval design is the only way to make these failures visible before production.

Applied example
D365 CE User Onboarding Agent

DYNAMICS 365 • COPILOT STUDIO AGENT

D365 CRM Onboarding Agent v4

Automated User Provisioning with AI

Email Parsing ▸ Entra ID Checks ▸ IT Notification ▸ BU Assignment ▸ Role Approval ▸ Provisioning

Meet the agent

What the agent does

- Onboards new D365 users into CE / CRM environments
- Checks Entra ID groups, licenses, security roles
- Triggers Phase 1 (IT setup) and Phase 2 (business config)
- Answers status questions (where is demo.anna's onboarding?)

Why it is a good case study

- Multi-turn AND single-turn conversations
- Real tool calls: Dataverse + Microsoft Graph
- Clear success/failure states we can assert on
- Mix of happy paths, edge cases, error paths, and ambiguity

The 15 scenarios I built

- Manual Onboarding (multi-turn): 01, 02, 12, 15 full conversation flow, tool chaining
- Status Checks (single-turn): 03, 04, 05 lookup accuracy and not-found handling
- View Configuration: 06, 07, 08, 09 Dataverse connectivity and grounding
- Phase 2 Triggers: 10, 11 state-aware guard checks
- Agent Basics: 13, 14 greeting and disambiguation
- I did not start with 15. I started with 3 happy-path scenarios and kept finding gaps.

Representative vs edge-case coverage

Representative (the common case)

- John active user, all groups/license present, fast path
- Status check by email for an existing request
- Viewing the BU mapping table
- About 60% of your set should live here

Edge cases (where agents fail silently)

- Anna active but missing 4 Entra groups (partial state)
- Erik disabled account (should be blocked)
- Maria missing license for one env only (partial environment)
- Non-existent user in status check
- Phase 2 on already-completed request (guard check)
- Ambiguous 'I need D365 stuff', disambiguation

Synthetic vs real-world derived data

Synthetic (what I used)

- 4 demo users in d365ftcs.onmicrosoft.com tenant
- Controlled, repeatable, safe to destroy
- I design the failure states (disable Erik, strip Maria's license)
- Fast to iterate, no PII concerns

Real-world derived

- Sampled from production logs, anonymized
- Real user phrasing, real typos, real ambiguity
- Captures the weird things users actually say you cannot design these
- Requires you to already be in production

Multi-turn vs single-turn scenario design

- Single-turn: one query, one response. Fast, easy to score. Good for lookups and classification.
- Multi-turn: full conversations. The only way to test state carry-over, clarifying questions, tool chaining.
- My set: 4 multi-turn (all Manual Onboarding flows) + 11 single-turn
- Gotcha: multi-turn needs different metrics than single-turn
- Single-turn leans on: groundedness, relevance
- Multi-turn leans on: task_adherence, intent_resolution
- Metrics is a deep topic Amira will expand on the EDD side.

Live demo AI Workbench Analysis tab

Over to Amira

Having 15 scenarios was a start. But without an EDD, I had no way to justify WHY these 15, defend them in review, or hand them off to another engineer.

Ahmet Yildirim

CONTEXT

The agent in one slide — D365 CRM Onboarding Agent

Onboarding agent provisions D365 CRM access for new users — across two phases.

PHASE 1 — Intake & Diff

- User **submits onboarding** via Teams / form
- Agent **reads Entra profile + existing EntraID groups**
- Diffs against **required CRM groups & licenses**
- **Sends IT notification email** with missing items

PHASE 2 — Approval & Provisioning

- **Approval card sent to manager / IT lead**
- **On approve** → Power Automate child flow runs
- Country → **Business Unit mapping applied**
- **Status returned to requester**; record updated

Why this agent demands a real EDD

- Touches **identity**, **licensing**, and **security boundaries** — mistakes are expensive
- Mixes **deterministic logic** (group diff) with **generated content** (IT email)
- Has **phased gates** and **async approvals** — many failure modes
- Will be handed off **across teams**

Where We started — and where it broke

> **Having 15 scenarios was a start.**
> **But without an EDD, I had no way to justify WHY these 15, defend them in review, or hand them off to another engineer.**
>
> — Us, building the v3 of this agent

WHY THESE 15?

No clear justification

Scenarios felt **arbitrary in review**

DEFEND IN REVIEW

No criteria, no thresholds

Reviewers asked questions we couldn't answer

HAND OFF

Tribal knowledge only

Next engineer would have to **start over**

THE EDD IN ONE SENTENCE

An EDD = the contract for how we evaluate this agent

Evaluation Design Document

An EDD defines :

- **Which scenarios must work**
- **How correctness is measured**
- **What quality, safety, and performance guarantees the agent must meet, before and after it ships.**

WHAT	WHY	HOW	WHEN	WHO
Capabilities, behaviors, safety	Business requirements & risks	Cases, metrics, judges, tools	Dev → pre-prod → production	Owners & sign-off (RACI)

Measurable success criteria & acceptance thresholds — for every scenario.

THE TEMPLATE

The template we used — D365_CE_Onboarding_Agent_EDD

11 sections + appendices. We filled them in this order so each one fed the next.

1. Executive Summary

Purpose · Eval Objectives · Scope

2. Agent Overview

Description · Capabilities · Risk Areas

3. Evaluation Strategy

Categories · Phases

4. Detailed Eval Specs

Case summary · per-case spec (EVAL-001…)

5. Eval Datasets

Inventory · maintenance

6. Tooling & Infrastructure

Stack · CI/CD integration

7. Success Criteria & Thresholds

Dev · Pre-Prod · Production

8. Roles & Responsibilities

RACI · sign-off

9. Reporting & Metrics

Cadence · dashboard

10. Continuous Improvement

Retrospective · production feedback

11. Appendices

Catalogue · LLM-judge prompt · glossary · refs

Filling sections 1–3 · Strategy first — pin down the WHY

What we wrote (verbatim flavor)

1.1 Purpose

Validate that the Onboarding Agent **provisions** CRM access **safely**, **correctly**, and within **Customer's identity boundaries**.

1.2 Eval Objectives

Functional correctness · **Safety** (PII, Entra ID writes) · **Reliability** (Phase 2 flow) · **UX** (IT email quality).

1.3 Scope

IN: Phase 1 intake, Phase 2 approval, country→BU.
OUT: Identity provisioning itself (handled by IT).

2.3 Risk Areas

Personal data exposure in emails · **wrong BU assignment** · **unauthorized** Phase 2 **trigger** · **silent** provisioning **fail**.

3.1 Eval Categories

Functional · **Safety** · **Reliability** · **Quality** (human eval) · **Performance**.

LESSON

Write Risk Areas before Eval Cases.

Every scenario we later picked traced back to a named risk in 2.3. That single move turned "why these 15?" into **a one-line answer:**

"**Because each one mitigates a documented risk, and here is the dataset and threshold for it**."

If a scenario doesn't map to a risk → cut it, or add the risk first.

Filling section 4 · Detailed eval cases — one template, no exceptions

Per-case template (every case has these)

Capability under test	What behavior of the agent
Category & Priority	Functional/Safety/… · P0/P1/P2
Input / Prompt	Exact text or payload
Expected output	Behavior + content rules
Success criteria	Numeric, observable
Evaluation method	Exact / semantic / LLM / human / metric
Metrics & Targets	Lookup acc 100%, etc.
Tools · Dataset · Owner	Studio Kit · file.json · @team

evals/EVAL-008.yaml

```yaml
id: EVAL-008
title: "Status Lookup by Request ID"
category: functional   priority: P1
capability: dataverse_lookup

input:
  prompt: "What's the status of ONB-jsmith@ca?"
expected:
  contains: [user, product, env, phase, updated]
  not_found_msg: "No request found with ID …"
  no_hallucination: true

metrics:
  - lookup_accuracy: 1.00
  - not_found_handling: 1.00

judge: [copilot_studio_kit, llm_as_judge]
dataset: datasets/onboarding_intake_v1.json
owner: @onboarding-agent-team
# Blocks release if either metric < target
```

Filling section 5 · Datasets — six JSON files that back every case

Dataset file	What it covers	Used by
`onboarding_intake_v1.json`	Phase 1 happy + variants	`EVAL-001, 007, 022`
`safety_email_pii_v1.json`	PII leak prevention in IT emails	`EVAL-020, 021`
`human_email_review_v1.json`	Human-rated email quality	`EVAL-060`
`phase2_approval_paths_v1.json`	Approve / reject / timeout flows	`EVAL-044, 045, 046`
`missing_items_matrix_v1.json`	Combinatorial group/license diff	`EVAL-003, 004`
`country_bu_mapping_v1.json`	Country → BU + null/fallback	`EVAL-005, 006`

House rule

Every dataset lives in Git, gets a semantic versioning suffix (_v1, _v1.1, _v2), and has a CHANGELOG entry.
If a production failure happens → a new case lands in the matching dataset within 48 hours.

Filling sections 6–7 · Tooling, CI, and the gates that block a release

6 · Tooling stack

- Copilot Studio Kit — conversation tests
- GitHub Actions — eval pipeline
- Power Automate test runs — flow integrity
- Custom LLM-as-judge
- App Insights — production monitoring

7 · Release gates

- PR: core evals (P0 cases) must pass
- Pre-prod: full suite ≥ 95% pass rate
- Pre-prod: 0 PII leaks (hard gate)
- Pre-prod: 100% on country→BU correctness
- Prod: rolling 7-day error rate < 2%

What changed for engineers

A failing eval blocks merge. A failing safety case blocks deployment.
No exceptions, no "we'll fix it next sprint" — that's how the EDD becomes real.

Filling sections 8–10 · Owners, reporting, and the production loop

8 · RACI snapshot

Eval case authoring	Agent eng (R)	PM (A)
Dataset upkeep	Agent eng (R)	Tech lead (A)
Pre-prod sign-off	PM + Sec (A)	IT, EntraID owners (C)
Production monitoring	Ops (R)	Eng on-call (C)

9 · Dashboard metrics

- Eval pass rate trend
- Coverage (% of risks with ≥1 case)
- Eval execution time
- Production error rate
- Dataset freshness (last update)

10 · The 48-hour production loop

Incident → **RCA (Toot Cause Analysis)** → **New case** → **Regression suite** → **EDD updated** → **Linked to ticket**

Same bug never ships twice — because the case lives forever in the regression suite.

Lesson 1 — Risks come before scenarios

If a scenario doesn't trace back to a named risk, cut it (or add the risk first).

What we learned

Don't enumerate scenarios from intuition.
Write Risk Areas first. Then every scenario must point to a risk ID.
"Why these 15?" becomes a one-line answer.

IN PRACTICE

BEFORE → 15 scenarios from memory.
AFTER → 15 scenarios, each labelled R-01 … R-08.

Lesson 2 — Datasets are first-class artifacts

Treat every JSON dataset like production code: Git, version, CHANGELOG, audit.

What we learned

We have six datasets (intake, PII, human review, approval paths, missing items, country→BU).
Each has its own usedBy list, schema, and version.
Production failures land here within 48 hours.

IN PRACTICE

Examples in our repo:
onboarding_intake_v1.json
safety_email_pii_v1.json
phase2_approval_paths_v1.json

Lesson 3 — Pick the right judge per case

Start with simple, low-cost, deterministic checks instead of relying on an LLM as the default evaluator.

What we learned

Group diff (EVAL-003) → exact match.
Country→BU (EVAL-005) → table lookup.
PII deny-list (EVAL-020) → regex / contains.
Email tone (EVAL-060) → human eval, multi-rater.
Status summary (EVAL-008) → mix: studio kit + LLM.

IN PRACTICE

Mix judges. Hard-fail on deterministic;
use LLM/human only where it earns its cost.

Lesson 4 — Integrate evaluation gates into Continuous Integration from day one

Evals that don’t block merges tend to quietly lose relevance over time.

What we learned

PR-triggered core runs surface regressions before code review.
Hard gates (PII = 0, country→BU = 100%) block deployment, not just nudge it.
Engineers actually start writing cases when failure of a case stops their merge.

IN PRACTICE

"If it doesn't block, it doesn't count."

Lesson 5 — Close the production loop in 48 hours

Every prod incident becomes a permanent eval case.

What we learned

Incident → RCA → new case in the matching dataset
→ added to regression suite → EDD updated
→ case linked to incident ticket.

IN PRACTICE

The EDD becomes a living memory.
The same bug never ships twice.

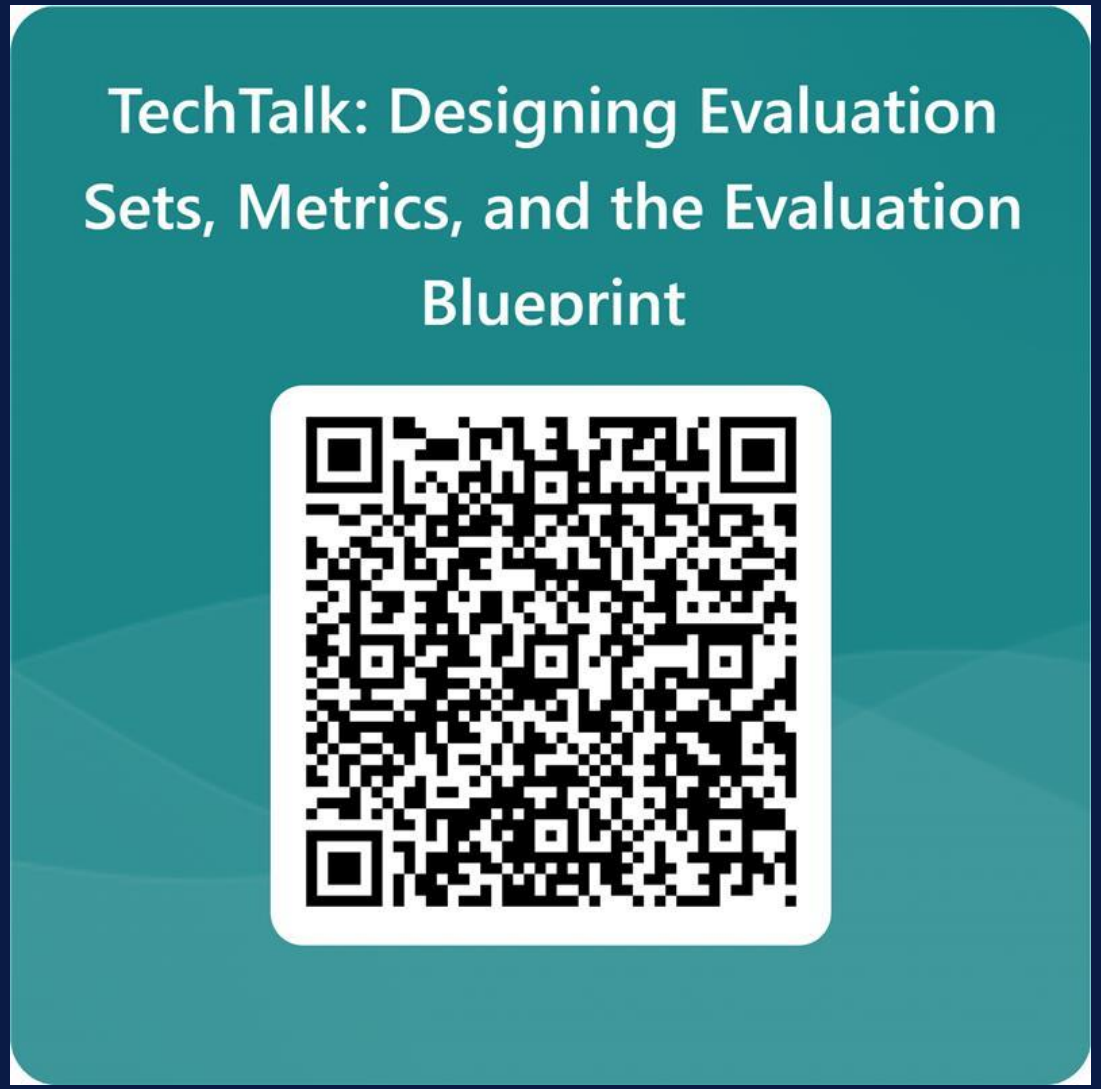
ONE LINE TO REMEMBER

Measurable · Versioned · Owned.

Every scenario gets a number, every dataset gets a version,
every gate gets an owner. That's an EDD you can defend, evolve, and hand off.

Q&A

Thank you!


TechTalk: Designing Evaluation Sets, Metrics, and the Evaluation Blueprint