From Layers to Answers

A five-step architecture for liberating geospatial data from its user interface — with FEMA's RAPT as the proving ground

JEFF FRANZEN · JUNE 11, 2026 · FOR DISCUSSION — INDEPENDENT RESEARCH

FEMA's Resilience Analysis and Planning Tool (RAPT) holds some of the richest public geospatial data in emergency management — hospitals, schools, nursing homes, demographics, hazard layers. Yet most of its value is locked behind a complex Experience Builder interface that forces users to think like GIS analysts: find the widget, select the layer, set the buffer, read the table. The data is excellent. The head attached to it is the bottleneck.

The Thesis

Stop building applications and start building answer infrastructure. Layers go in; a documented catalog and a library of analytical recipes sit in the middle; any interface comes out. Two intelligence steps — one that captures what the data can answer, one that executes those answers on demand — replace the training sessions, the widget hunts, and the dependence on a GIS specialist for every question. The user asks in plain English and receives a complete, sourced answer, rendered in whatever interface fits the moment: Experience Builder, a dashboard, a custom map, or no interface at all.

The Old Way Stops at Step One

The standard GIS delivery pattern has three moves: acquire data, publish layers, and assemble those layers into an application — then tell the user to figure it out. Every step after publishing happens inside the user's head. Which layers answer my question? In what sequence? With what analysis? That mental work is why every Experience Builder deployment needs a training session, and why most users ask a GIS person instead of using the tool.

Independent research by a Red Cross volunteer. Not an official American Red Cross or FEMA application or position.

The Five Steps

The new architecture moves the analyst's thinking out of the user's head and into the pipeline. Two intelligence steps sit between the data and the screen.

1 DATA

Layers

Source, clean, and publish feature layers. Unchanged from today — RAPT's existing AGOL and REST services already do this well.

2 INVENTORY

Catalog

Layers become documented, queryable API endpoints rather than ingredients dumped into an app: names, schemas, fields, refresh cadence, and what each layer contains. RAPT is treated not as a map but as a catalog of services.

3 INTELLIGENCE · KNOWLEDGE

Ideas

The semantic layer. For each catalog entry: what questions can it answer, and through which analytical recipes — buffer + intersect + demographic rollup, storm path × nursing homes, drive-time × facility placement. This is the GIS analyst's judgment, written down as machine-readable intent.

4 INTELLIGENCE · EXECUTION

AI

An agent reads the catalog and the ideas, takes a plain-English question, and orchestrates the function calls — geocode, spatial query, demographic lookup — to produce a complete answer with sources.

5 OUTPUT

Any UI

The head detaches from the body. One backend serves Experience Builder, a dashboard, a custom Mapbox app — or no interface at all: a headless service where the user simply asks questions.

The distinction between steps 3 and 4 is what makes this five steps rather than four. The catalog tells the agent *what exists*; the ideas layer tells it *what is possible and how*. Without step 3, the agent rediscovers GIS reasoning from scratch on every question — and gets it wrong often enough to lose the user's trust. With it, the agent executes proven recipes.

What a Recipe Looks Like

An ideas-layer entry is small, concrete, and reusable. It names the question, the catalog entries involved, and the sequence of operations — the exact knowledge a GIS analyst would otherwise apply by hand, session after session.

> *"Which nursing homes are in the storm path, and do they have backup power?"*
>
> | | |
> |---|---|
> | LAYERS | `hifld/nursing_homes` · `nws/storm_track_cone` · facility attributes (`backup_power`) |
> | STEPS | 1. Fetch current storm cone geometry → 2. Spatial intersect against nursing-home points → 3. Join facility attributes → 4. Flag records where backup power is absent or unknown |
> | OUTPUT | Ranked list with counts, map geometry for any UI that wants to draw it, and source citations per record |
> | CAVEATS | Cone ≠ impact area; attribute vintage varies by state — the agent says so in the answer |

Multiply this by a few dozen recipes — wildfire perimeter × census tracts, flood gauge × shelter locations, drive-time × trailer placement — and the ideas layer becomes the institution's geospatial judgment, captured once and executed on demand. New recipes are added the way lessons learned should be: when a question gets answered well in the field, the recipe is written down.

What Changes for the User

	OLD WAY — APP-CENTRIC	NEW WAY — ANSWER-CENTRIC
User thinks like	A GIS analyst: widgets, layers, buffers, attribute tables	A decision-maker: "Which nursing homes are in the storm path, and do they have backup power?"
Where analysis lives	In the user's head, every session, from scratch	In the pipeline — captured once in the ideas layer, executed by the agent
Interface	One fixed application; the UI *is* the product	Any UI — Experience Builder, dashboard, Mapbox, or headless chat; the answer is the product
Training burden	A session per app, repeated per audience	None — plain English is the interface
Maintenance	Fighting widgets in a 2015-era builder	Managing logic: catalog entries and recipes

Independent Validation: Perplexity's "Search as Code"

This architecture is not a one-off intuition. Perplexity's search research reaches the same conclusion in a different domain: give the agent a small set of fundamental primitives, and let it *compose them in code* rather than choosing from pre-built workflows.

> *The SDK provides the most fundamental primitives, and the model can build any additional components on the fly with code.*
>
> — PERPLEXITY RESEARCH, "RETHINKING SEARCH AS CODE GENERATION"

Their finding maps one-to-one onto geospatial work. Without composition, an agent fires a single approximate query, filters a noisy result, and hopes. With it, the agent writes a small program — parallel calls, exact predicates, precise rollups — and returns the complete answer. Perplexity applied the pattern to information retrieval; the five-step model applies it to spatial analysis. The catalog and ideas layers are the geospatial equivalent of their SDK primitives and documentation: the substrate that makes agent-written orchestration reliable instead of improvised.

Why RAPT Is the Right Proving Ground

RAPT is public, authoritative, and already API-shaped — every layer is exposed as an ArcGIS REST service. No data migration and no agency permission are needed: the transformation happens entirely in steps 2 through 5, on top of services FEMA already publishes. A working demonstration — a documented catalog of RAPT endpoints, a small ideas file of proven recipes, an agent answering live questions — proves the thesis on the most public dataset available, and the pattern then transfers directly to Red Cross operational data, where the layers and enrichment pipelines already exist.

The Ask

Treat this as an architecture discussion, not a product pitch. Three questions worth debating: Should new GIS data products ship as catalog entries plus recipes rather than finished apps? Who curates the ideas layer? And which dataset — RAPT or internal operational layers — makes the most convincing first demonstration?

The one-sentence version: stop building applications and start building answer infrastructure — layers in, catalog and recipes in the middle, any interface out.