From MCP Servers to Local Skills: A Token-Efficient, Model-Agnostic Architecture for an LLM Assistant in Fire and Rescue Services

1st Maroun Assaf
ENSTA Paris
Internship at FEMTO-ST Institute
Palaiseau / Besançon, France
maroun.assaf@ensta.fr

2nd Christophe Guyeux
FEMTO-ST Institute, UMR 6174 CNRS
Université Marie et Louis Pasteur
Besançon, France
christophe.guyeux@univ-fcomte.fr

***Abstract*—Fire and rescue services need decision support that pulls together many live data sources, among them weather, fire-danger indices, flood vigilance, operational doctrine and water resources, under tight latency and cost budgets. This paper reports on PredictOps, an assistant built for the French departmental fire and rescue services (SDIS). Its first generation paired a set of Model Context Protocol (MCP) servers with a cloud-hosted multi-agent workflow: each question was classified and routed to a specialised agent, which in turn called the relevant tools. The design worked, but it was slow and expensive, since every request paid for several sequential large language model (LLM) turns and a string of network hops before the user saw anything. We describe how we migrated the system to local skills, that is, self-contained command-line tools that the model runs directly and whose structured output it reads, dropping both the intermediate agents and the MCP round-trips. On a ten-topic doctrine benchmark the resulting skill answered every question in a single LLM turn, at an average of 10.3 s and $0.0117 per query, against tens of seconds and eight to sixteen turns for the agentic design it replaced. We then wrap those same skills as tools of a model-agnostic service built on LangChain, so that execution is portable across providers and gated by role-based access control. We also describe the eighteen skills currently deployed, the public data sources they call, the operational framing that justifies them, and the technical obstacles we had to clear.**

***Index Terms*—large language models, tool use, Model Context Protocol, emergency services, decision support, retrieval-augmented generation, access control, local computation**

I. Introduction

French departmental fire and rescue services, the *Services départementaux d'incendie et de secours* (SDIS), plan and run emergency response on information scattered across dozens of public and internal systems. An officer preparing an intervention might need the local forecast and any official vigilance alert, the fire-danger level of a nearby forest, the flood status of a river, the closest fire-fighting water points, the doctrine that applies to the situation at hand, and demographic or industrial-risk data for the area. Every one of these sits behind its own interface, and none of it helps unless it can be pulled together quickly.

PredictOps places all of it behind a single conversational assistant. A user asks a question in plain language; the assistant fetches the relevant data, reasons over it, and answers in French, with sources. The first production version rested on the Model Context Protocol (MCP) [1] and a multi-agent workflow authored in a visual builder, delivered to the user through a chat interface. This paper describes that assistant and then explains why we walked away from that architecture. Our argument is that for this class of application, MCP servers fronted by a separate cloud agent that orchestrates them are both slow and costly: each request pays for several sequential LLM inferences and several network round-trips before an answer appears. We replaced the arrangement with a *pure-skill* design, in which the model itself runs a local tool and reads its output, and we made the whole system model-agnostic by rebuilding the orchestration layer on LangChain [2].

The contributions of this paper are the following. We report an industrial migration from a tool-server architecture to local skills, with measurements from a real deployment rather than a synthetic benchmark. We describe a catalogue of eighteen skills covering the operational needs of a fire and rescue service, organised by family. We document a set of technical obstacles, several of them specific to retrieval quality and to cross-platform execution, together with the fixes that worked and the ones that did not. Finally, we show how the same skills are made provider-independent and role-aware by a thin LangChain layer.

The paper proceeds as follows. Section II situates the work. Section III describes the assistant and its operational value. Section IV lists the skill catalogue by family. Section V presents the original multi-agent MCP workflow. Section VI explains the move to skills, and Section VII the choice of LangChain for the model-agnostic layer. Section VIII reports the technical problems and their resolution, Section IX discusses the operational framing and the limits, and Section X concludes.

II. Related Work

Giving a language model access to external tools is now a standard way to ground its answers. Early work showed that a model can learn when to call an API and how to splice

the result back into its generation [8], and that interleaving reasoning steps with tool calls improves both accuracy and traceability [7]. Scaling this to large tool catalogues has been studied through retrieval over API documentation [9] and through controller models that dispatch to specialised back ends [10]. Multi-agent frameworks generalise the pattern by letting several conversing agents divide the task [11], and recent surveys map the resulting design space [12]. Our experience is a counterpoint to that trend: for a bounded, well-understood set of domains, adding agents mostly adds sequential inferences, and the decisive factor turned out to be retrieval quality rather than orchestration depth.

Grounding answers in a document corpus is the province of retrieval-augmented generation [3], which remains the accepted remedy for the factual errors that language models produce when left to their own parameters [13]. We use a classical lexical ranking function, BM25 [4], rather than a dense retriever, for reasons of latency, determinism and offline operation that we detail in Section VI. The Model Context Protocol [1] standardises how a model reaches a tool server, and our criticism concerns the deployment shape it encourages when the tools are cheap and local, not the protocol itself.

In the emergency and health domain, conversational assistants have been evaluated in hospital settings, where a chatbot answering caregivers' questions about drugs and pharmacy organisation around the clock proved feasible and was well received [14]. Sovereign assistants aimed specifically at French firefighters are also appearing [6]. Domain models such as the Canadian Fire Weather Index [5], adopted across Europe for fire-danger rating [15], and the national open-data portals for hazards, water and addresses, supply the substrate that our skills compute over. Access control follows the classical role-based model [16].

Our position differs from most of this literature on one point: the system is about to leave the laboratory. Published assistants of this kind are usually evaluated on question panels, whereas ours is being prepared for a first operational deployment with the firefighters of Belfort, with whom discussions are under way, so that real cases rather than curated ones drive what we learn. That prospect shaped the engineering reported here more than any benchmark did. Access control is enforced deterministically instead of by prompt, and every answer carries its sources, because a user who acts on an answer will be held responsible for the action. For the same reason, the execution layer was made independent of any single model provider before the first user ever saw it.

III. The PredictOps Assistant

A. What it does

The assistant answers operational questions by calling, on demand, a set of tools that each cover one domain. A single question often needs several of them at once. Asked to prepare an intervention at a given address, the assistant geocodes it, pulls the weather and any vigilance alert, lists the nearest water points and their distances, reports the industrial and natural risks and the flood vigilance of the surroundings, checks the terrain and, if a wooded massif is close, the forest-fire danger, and locates the nearest hospital with its real travel time by road. The reply comes back as a structured briefing, and whenever it contains geolocated items the assistant also draws an interactive map.

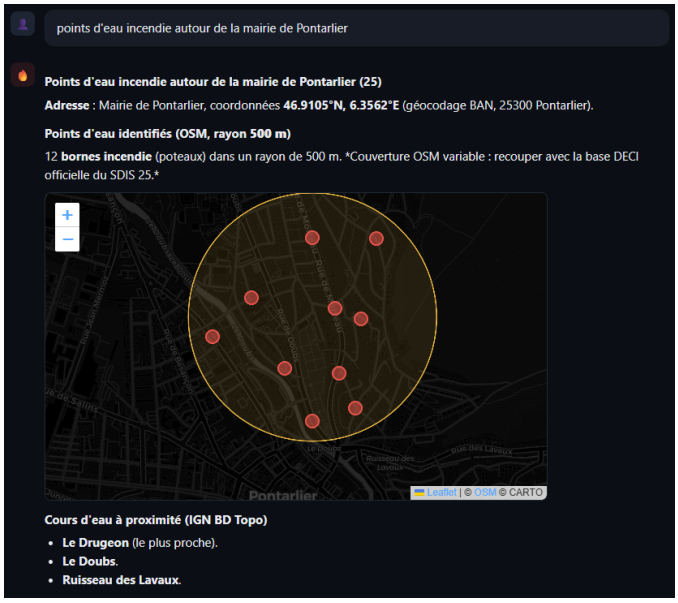


Fig. 1. A geographic answer. The assistant geocodes the requested address, calls the fire-water-point skill, and emits an interactive map block with the hydrants found in a 500 m radius, followed by the nearest watercourses usable for supply. Coordinates come only from the skills; the model is forbidden to invent them.

Figure 1 shows a typical geographic answer. The user asks for the fire hydrants around the town hall of Pontarlier; the assistant geocodes the address through the national address base, queries the water-point skill, and renders the twelve hydrants found within a 500 m radius on a map, followed by the watercourses nearby that could serve as a suction point. The answer states its source and warns that OpenStreetMap coverage is uneven and should be cross-checked against the official departmental DECI database, a caveat we return to in Section IX.

B. The departure sheet

The most complete operational artefact the assistant produces is the *fiche de départ*, a departure sheet compiled for an intervention address and shown in Figure 2. One request chains the geocoding, weather and vigilance, fire-water-point, hazard, terrain and hospital skills, and returns a numbered briefing that can be printed or saved as PDF directly from the browser. The sheet in the figure was produced for a street address in Belfort: the geocoding resolves to house-number precision with a confidence of 0.97, the weather block gives the current conditions and the next day's yellow heatwave vigilance issued by Météo-France together with a moderate fire-danger appraisal, and the water block reports twenty-four hydrants within 600 m, the five nearest listed with their distance and network. An earlier run on an address

TABLE I
THE EIGHTEEN DEPLOYED SKILLS, BY FAMILY

Family	Skills and coverage
Weather and fire danger	`meteo` (current, forecast and archive since 1940, plus the official Météo-France vigilance level); `meteo-forets` (Fire Weather Index computed locally, 7-day outlook)
Water and flood	`vigicrues` (official real-time flood vigilance, 337 monitored reaches); `environnement` (river stage and discharge, groundwater, water and air quality); `points-eau-incendie` (hydrants, suction points, tanker autonomy and shuttle cadence)
Location and mobility	`geocodage` (address to coordinates, reverse geocoding, nearby points of interest); `itineraire` (road distance and travel time, multi-origin comparison)
Terrain and land use	`terrain` (forest stands, road network and gauge limits, forest-fire tracks, buildings, watercourses); `agriculture` (crop parcels, buffer strips, sensitive grassland, wetlands)
Risk and territory	`risques-territoire` (flood, seismicity, radon, clay shrink-swell, cavities, classified and Seveso sites, disaster history); `territoire-insee` (population, vulnerability, income, housing, facilities)
Doctrine and law	`doctrine-search` (BM25 search over the operational doctrine corpus); `legifrance` (codes, statutes, case law, constitutional decisions)
Health and population	`epidemiologie` (Sentinelles incidence and trends); `attente-urgences` (emergency-department waiting times); `geodae` (national defibrillator database)
Internal platform	`interventions` (volumes, peak hours, seasonality, year-on-year comparison); `supervision` (call-handling quality, model explainability, daily feature store)

in Besançon returned the same five sections from live data and a summary map carrying seven markers in three semantic colours.

C. Why it is useful operationally

The assistant collapses a manual, multi-source lookup into one request answered in seconds, in the user's own language, with the data already cross-referenced. Officers need not know which portal holds which dataset, or how to query each one. Because answers cite their sources, and the system is instructed never to invent coordinates or legal text, the output stays auditable, as a safety-critical setting requires. Access is scoped by professional profile, so each user sees only the tools relevant and permitted for their role. A task that used to mean several minutes of navigation across separate systems becomes a single question and a single sourced answer.

IV. THE SKILL CATALOGUE

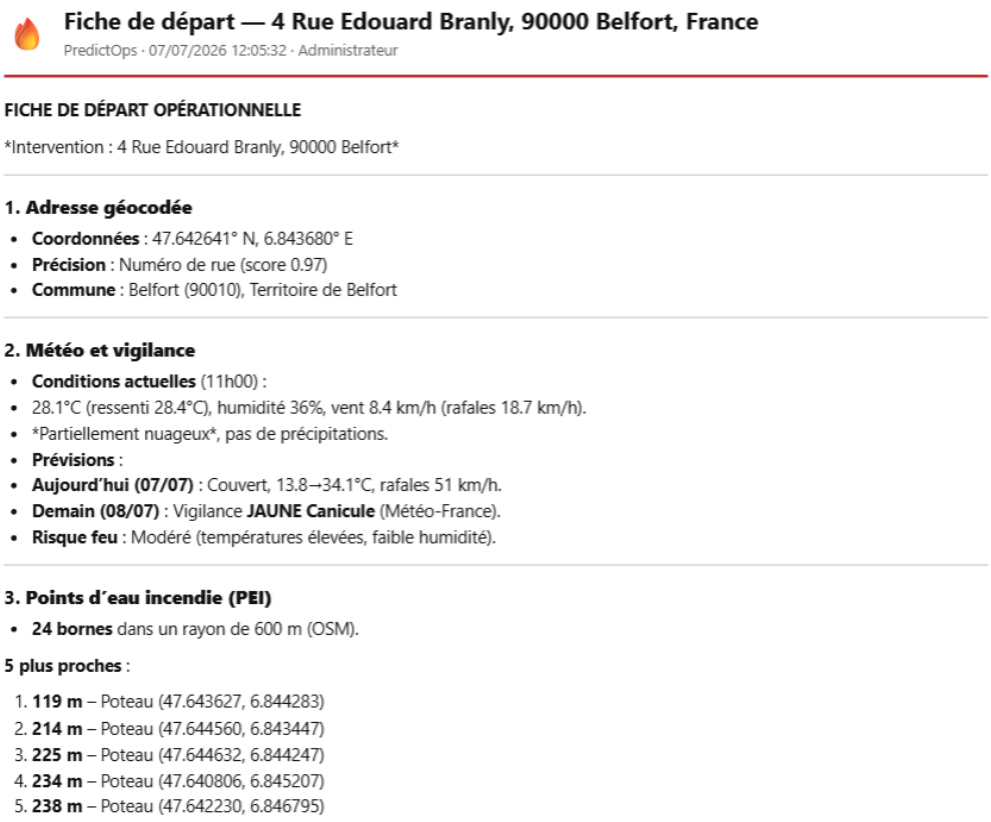
Eighteen skills are currently deployed. Fourteen of them come from the migration and cover roughly thirty-one of the forty tools that had been wired into the original workflow; the remaining four were written from scratch for needs the original system never addressed. Table I groups them into eight families.

TABLE II
PUBLIC SERVICES AND INTERNAL SOURCES CALLED BY THE SKILLS

Source	Endpoint and use
Open-Meteo	api.open-meteo.com/v1/forecast, archive-api.open-meteo.com/v1/archive: forecast and hourly archive back to 1940, also the input series of the fire-danger computation
Météo-France	public-api.meteofrance.fr/public/DPVigilance/v1: official departmental vigilance bulletin (key required)
Vigicrues	www.vigicrues.gouv.fr, InfoVigiCru GeoJSON service: real-time flood vigilance colours and monitored reach geometries
Hub'Eau	hubeau.eaufrance.fr: hydrometry stations and observations, groundwater levels, river physico-chemical quality
Géod'Air	www.geodair.fr/api-ext: air-quality measurements and statistics (key required)
Géorisques	www.georisques.gouv.fr/api/v1: flood zones, seismicity, radon, clay shrink-swell, cavities, classified and Seveso sites, disaster history
IGN Géoplateforme	data.geopf.fr/geocodage/search and /reverse for addresses, data.geopf.fr/wfs/ows for BD TOPO, BD Forêt and RPG feature queries
Base Adresse Nationale	api-adresse.data.gouv.fr/search, geo.api.gouv.fr/communes: address and commune resolution
Overpass (OSM)	overpass-api.de/api/interpreter with three mirrors: hydrants, suction points and nearby features
OSRM	router.project-osrm.org: road distance and travel time, multi-origin comparison, route geometry for the map
Légifrance	api.piste.gouv.fr/dila/legifrance/lf-engine-app through oauth.piste.gouv.fr: codes, statutes and case law (key required)
Sentiweb	www.sentiweb.fr/api/v1/datasets/rest: Sentinelles network incidence series
data.gouv.fr	www.data.gouv.fr/api/1/datasets: dynamic resolution of the national defibrillator dataset
Internal	MongoDB `pompier` database (interventions, call quality, waiting times, feature store); local INSEE parquet files; local doctrine corpus

None of these skills owns its data. Each one queries a public service, an internal database or a local corpus, and the value it adds is the operational framing of the answer rather than the raw record. Table II lists the services actually called by the deployed code. Almost all of them are French or European public open data, reachable without a key; only three require credentials, namely the Météo-France vigilance bulletin, the Légifrance engine behind the PISTE gateway, and the Géod'Air air-quality export. Internal data, that is intervention volumes, call-handling quality, emergency-department waiting times and the daily feature store, comes from the platform's own MongoDB instance, and the territorial indicators are read from local INSEE parquet files.

Two design rules govern the catalogue. First, related tools are deliberately grouped into a small number of thematic skills rather than one skill per original tool, because every manifest loaded into context costs tokens; this is why about forty wired MCP tools condense into eighteen skills, and why a single skill such as `environnement` absorbs four former servers. Second, each skill must be generic over its data rather than



Fiche de départ — 4 Rue Edouard Branly, 90000 Belfort, France

PredictOps · 07/07/2026 12:05:32 · Administrateur

FICHE DE DÉPART OPÉRATIONNELLE

Intervention : 4 Rue Edouard Branly, 90000 Belfort

1. Adresse géocodée

- **Coordonnées** : 47.642641° N, 6.843680° E
- **Précision** : Numéro de rue (score 0.97)
- **Commune** : Belfort (90010), Territoire de Belfort

2. Météo et vigilance

- **Conditions actuelles** (11h00) :
- 28.1°C (ressenti 28.4°C), humidité 36%, vent 8.4 km/h (rafales 18.7 km/h).
- *Partiellement nuageux*, pas de précipitations.
- **Prévisions** :
- **Aujourd'hui (07/07)** : Couvert, 13.8→34.1°C, rafales 51 km/h.
- **Demain (08/07)** : Vigilance **JAUNE Canicule** (Météo-France).
- **Risque feu** : Modéré (températures élevées, faible humidité).

3. Points d'eau incendie (PEI)

- **24 bornes** dans un rayon de 600 m (OSM).

5 plus proches :

1. **119 m** – Poteau (47.643627, 6.844283)
2. **214 m** – Poteau (47.644560, 6.843447)
3. **225 m** – Poteau (47.644632, 6.844247)
4. **234 m** – Poteau (47.640806, 6.845207)
5. **238 m** – Poteau (47.642230, 6.846795)

Fig. 2. Departure sheet generated for an intervention address. A single composite request chains the geocoding, weather and vigilance, and fire-water-point skills; the result is a numbered briefing, printable to PDF, with coordinates, precision score, current conditions, the vigilance bulletin, the fire-danger appraisal and the nearest hydrants ranked by distance. Shown at full text width, rather than column width, for legibility.

over its code. The doctrine engine, for instance, discovers any corpus directory matching a naming pattern and federates the search across all of them, merging by score and tagging every hit with its department, so that adding a new departmental corpus requires no change to the code or to the manifest.

V. The Original Multi-Agent MCP Workflow

The first production assistant had two layers. The tools themselves lived as MCP servers, small services, one per domain, each wrapping a public API or an internal database and exposing it as a handful of callable tools. Above them sat an orchestration workflow authored in a visual agent builder and served to the browser through a chat runtime.

That workflow was a tree of agents. An incoming message reached a classifier whose sole task was to file it under exactly one category. Depending on the category, control passed to further classifiers, a dedicated one for territorial-statistics questions for instance, and then to a specialised agent for the domain: weather, terrain, housing, facilities, economy, demography, epidemiological bulletins, air quality, and so on. The chosen agent carried the instructions and tool bindings for its domain and issued the MCP calls needed to answer. So that follow-ups like "yes" or "a pie chart" would still classify correctly, the classifier saw the last few exchanges rather than the latest message alone.

The scheme was expressive and easy to grow one domain at a time, but its economics were poor. A single question could set off a chain of LLM turns, one or more to classify and at least one for the specialised agent, before a single tool was touched, and each MCP call was its own network hop from the backend to the tool server. Latency and token consumption thus rose with the depth of the routing tree, and every request paid that toll no matter how trivial the underlying lookup.

VI. From MCP to Skills

Our goal in the migration was to strip out the intermediate orchestration and the network hops while leaving the domain logic untouched. The pattern we settled on, a *pure skill*, swaps the pairing of an MCP server and a separate orchestrating agent for local computation the assistant runs itself. A skill is just a directory: a manifest saying when to use it, plus a self-contained command-line script written against the standard library. When a question matches, the model runs the script, reads its structured output, and writes the answer. There is no second agent, no intermediate inference and no long-lived tool server.

A. The doctrine case study

We proved the pattern first on operational doctrine, the most token-hungry tool in the original system. The corpus for a

single department holds 1565 indexed pages, each carrying a keyword block, the OCR of the page body and the OCR of its figures, where the dosage tables live. The doctrine skill indexes it with a BM25 ranking function [4] implemented entirely in the standard library, with no network access, then retrieves the pages relevant to a query and hands them to a single model inference that writes a sourced answer. All measurements reported in this section, the agentic-to-one-shot cost and latency progression below and the benchmark of Table III, use the same model throughout, Mistral Large (`mistral-large-latest`), chosen for its EU-hosted inference given the sensitivity of operational fire and rescue data.

Two findings shaped the final design. First, answer quality hinged far less on orchestration than on retrieval. On the query "wasp sting", literal BM25 returned an operational sheet about Asian hornets as its top hit, while the medical pages on anaphylaxis were absent from the ranking altogether. A small curated table of domain and clinical synonyms, applied at query time with a reduced weight and no re-indexing, brought those pages up to ranks two through five. The effect on the agentic version was immediate: sixteen turns fell to eight, cost fell from \$0.14 to \$0.07, and latency from 76 s to 53 s. The value the autonomous sub-agent had been adding was query reformulation, not reasoning.

These two effects, the retrieval fix and the later architecture change, are not equal in size, and crediting the full reduction to the migration alone would overstate the architectural contribution. Decomposing the two-step progression on the figures already given, \$0.14 to \$0.07 with the synonym table and an otherwise unchanged agentic architecture, then \$0.07 to the \$0.0117 mean of Table III after the move to the one-shot design below, the retrieval fix accounts for 55% of the total cost reduction and the architecture change for the remaining 45%. The same decomposition on latency, from 76 s to 53 s and then to the 10.3 s mean of Table III, gives 35% to retrieval and 65% to the architecture. Retrieval quality and the architecture change therefore each account for a substantial share of both gains, with neither dominating as sharply once the figures are grounded in the reproduced benchmark: a more accurate summary than attributing either reduction to a single cause.

Second, latency was dominated by the agentic loop of sequential LLM turns, not by the model call itself. Moving to a one-shot design, in which the launcher performs the retrieval locally and then calls the model once with no tools at all, cut a typical query from the 53 to 85 s range down to a few seconds to under twenty, the upper end now bounded by generation rather than by round-trips. Table III reports a ten-topic benchmark of that configuration, rerun on 2026-08-03 against the current corpus and published in full, script and raw per-call output, including retrieved sources, token counts and answer text, in the `skills_paper_eval` directory of the project repository[1], since the original development-time measurements could not be recovered from any artifact. Each of the ten questions was run three times to quantify run-to-run variance, which was modest for most questions (coefficient of variation under 30%) but reached 94% on the shortest, most templated answer, where a few seconds of generation time swings the ratio sharply even though the answer itself is stable across runs.

TABLE III
ONE-SHOT DOCTRINE SKILL, TEN-TOPIC BENCHMARK (30 CALLS, 10 QUESTIONS × 3 RUNS, MISTRAL LARGE, RERUN 2026-08-03)

Metric	Value
Success	30 / 30 (95% Wilson CI: 88.6–100%)
Fallbacks to the agentic path	0
Mean latency	10.3 s (median 8.9 s, range 1.8–19.0 s, s.d. 4.5 s)
Mean tokens	3909 prompt + 640 completion = 4549 total
Mean cost	\$0.0117 per query (range \$0.0064–\$0.0173)
LLM turns	1, in every case

The one-shot figures quoted throughout this paper, in the abstract, in this section and in the conclusion, are this reproduction's own numbers rather than the original development-time measurements, whose log could not be recovered from any artifact (Section IX); we chose to report a single consistent set of figures rather than mix a recovered reproduction with unrecoverable originals. This reproduction fixes retrieval at five pages of full text per query, a configuration choice documented in the released harness and not necessarily identical to whatever depth produced the original figures during development; the qualitative pattern, a single LLM turn, sub-20 s latency and a cost under two cents per query, matches what was reported at the time.

A final step removed the launcher itself. Rather than have the backend spawn a one-shot model call as a subprocess, the deliverable became a skill the assistant executes directly: manifest plus BM25 script, nothing else. Retrieval, page reading and synthesis all happen inside the agent's own loop, which makes the iteration free and deletes the fallback ladder that the launcher had needed.

B. Why the switch pays off

The case for the migration follows from these measurements. Dropping the classifier tree and the intermediate agents removes the turns spent deciding what to do before doing it. Running the tool locally removes the hops to a separate MCP server. Retrieval and deterministic lookups, which are most of what the tools actually do, need no model at all and reduce to plain local computation; sizing the hydraulic autonomy of a tanker, for example, is arithmetic, and paying for an inference to perform it is waste. The payoff is lower latency and lower token cost, and the design scales, because the per-request overhead no longer climbs with the number of domains.

C. Three variables, not one

The comparison above is easy to misread as evidence against MCP itself, so it is worth being explicit about what

[1] https://github.com/cguyeux/predictops-mcp/tree/migrate-to-skills/skills_paper_eval, commit `e0db3ac9`.

it does and does not isolate. Three variables are at play, and they are conceptually independent: *transport*, whether a tool is reached over the network or run as a local process, noting that MCP itself runs as well over stdio as over HTTP and is not inherently remote; *orchestration*, whether a classifier tree of agents decides what to call or a single LLM turn does; and *retrieval quality*, whether the query reaching the ranking function is literal or synonym-expanded. The title and Related Work (Section II) already scope our criticism to the deployment shape we inherited, a remote tool server behind a separate cloud orchestrator, rather than to the protocol in the abstract, but the measurements in Section VI only isolate two of the three variables. The synonym-table fix alone moved the agentic path from sixteen turns and $0.14 to eight turns and $0.07 with the architecture otherwise unchanged, which isolates retrieval; the move from that eight-turn agentic path to the one-shot skill changes orchestration and transport together, since the tool call stops being both routed through a classifier and addressed over the network at the same time. A fourth configuration, an MCP server addressed locally over stdio by a single-turn caller, would isolate transport from orchestration, and we did not run it. We say so rather than let the architecture comparison imply a transport result it does not contain.

VII. Model-Agnostic Execution with LangChain

As first built, the pure-skill pattern assumed one particular model provider. An operational deployment cannot hang on a single vendor, and the fire and rescue services asked, in so many words, to be able to run other models. Hence the second change: a dedicated model-agnostic service, built on LangChain and its graph extension for agent control flow [2].

The service rewrites none of the domain logic. Since every skill already ships a self-contained command-line script, each one is wrapped automatically as a LangChain tool: a scanner reads the manifest front matter for the name and description, lifts the first block of shell examples to show the model the real calling convention, and runs the script as a subprocess with a bounded argument list, a 120 s timeout and truncated output. A graph-based agent is handed the tool list and a model chosen at run time through a single model-initialisation entry point, so the same skills run under different providers by configuration alone. A model whose API key is missing stays visible in the interface but greyed out, and turns on the moment the key is supplied.

Building on LangChain also handed us the surrounding machinery an operational deployment needs, without our having to reinvent it. Conversations, user accounts and profiles persist in a dedicated database, isolated from the business data, with transparent fallback to an in-memory store when the database is unreachable. The role attached to a user in that store, never a value sent by the client, determines which skills are exposed. Answers stream token by token over server-sent events, and the interface renders charts, diagrams and interactive maps that the model emits as fenced blocks, with popups inserted as text rather than markup so that a skill's output cannot inject HTML.

VIII. Technical Problems and Their Resolution

Several obstacles were not visible from the architecture and only surfaced in practice. We report them because most were silent failures rather than crashes.

Silently truncated corpora on Windows. Doctrine file paths reach about 470 characters, well past the legacy 260-character limit. Directory walks that are not prefixed to opt into long-path handling skip the deep files without raising anything, and the index reported 1409 documents where 1565 existed. The fix was to apply the extended-length prefix on every input and output operation and, critically, on the root of the directory walk itself. A related nuisance was that 53 archive entries produced on macOS carried a colon in their name, which is illegal on Windows; extraction now sanitises each path component.

Console encoding. Scripts printing arrows or accented characters aborted with an encoding error on the default Windows code page. Every skill script now reconfigures its standard output to UTF-8 as its first act, which is part of the skill pattern rather than a per-script patch.

Conversational noise in retrieval. After the synonym table was in place, a naturally phrased question still degraded badly while its terse equivalent did not. The culprit was filler words, not a missing synonym: French equivalents of "give", "me", "do" and the auxiliaries, absent from the stop-word list, matched unrelated pages, so a sheet about tablet errors outranked the correct medical pages and the character budget truncated them. Extending the stop-word list to conversational fillers fixed the class of problem; curated synonym tables proved too specific, and generic corrections to the token stream outperformed adding words one at a time.

False detection of abandonment. The launcher escalated to a larger model whenever the answer contained a marker phrase indicating insufficient extracts. A correct, complete answer that added an honest caveat using that phrase therefore triggered a needless and expensive escalation. Counting the marker only inside short answers, below a length threshold, distinguishes a genuine refusal from a qualified answer.

A hole in prompt-enforced access control. Refusal driven purely by the system prompt proved unreliable: a profile without doctrine access, asked for the conduct to hold on a wasp sting, still received an answer, because the model judged it did not need the tool and replied from memory, at once an access-control failure and a safety risk, since the answer was medical and unsourced. The remedy is defence in depth: a deterministic keyword classifier runs before the model and blocks a request that falls under a forbidden skill and under no permitted one, with accent- and punctuation-insensitive normalisation, and an anti-over-blocking rule that lets a request through when it also touches an allowed skill; the filtered tool list and the prompt-level refusal remain as the second and third layers. An unknown role receives no skills at all, so the system fails closed rather than sliding back to a broader profile.

Perceived latency. Total latency is bounded by generation, but the user experience is not. Introducing streaming moved

the first visible feedback from a mute spinner lasting the whole request to 4.2 s, with the tool call announced then, completed at 5.1 s, the first answer token at 9.0 s and completion at 14.9 s. The total time did not change, but the wait stopped looking like a hang. The streaming generator is deliberately synchronous so that the framework iterates it in a thread pool, avoiding any mixing of an event loop with synchronous database access.

Large reference datasets. The national defibrillator database is an 82 MB file of 181 366 records whose download URL changes daily and whose tabular API is withdrawn. The skill resolves the resource dynamically through the dataset API and rebuilds a compact fifteen-column local cache by streaming, with a seven-day time-to-live, so that a proximity query costs about two seconds against a bounding box followed by a haversine refinement, and never a fresh download.

Getting at the data on the production server. The hardest obstacle was not in the code at all. Several skills read reference files that live neither in the repository, which excludes them by design, nor on the production host we can reach: an audit confirmed its `data` directory holds only a few legacy folders, its doctrine directory is empty, and its checkout is more than five months behind the development branch. The files sit on a colleague's workstation and reach the server through a manual synchronisation script. Volume is the real difficulty: the remaining departments' doctrine corpora alone weigh tens of gigabytes, one about forty-five, and the IGN, IRIS and INSEE layers add tens more, on a host already ninety-five per cent full. No engineering trick avoids the copy. What the code can do is degrade honestly instead of silently: the doctrine engine discovers whichever corpora are present and reports which it searched, so a partially synchronised deployment returns fewer sources instead of wrong ones, and each skill fails with an explicit message naming the missing dataset. Only the database needed no transfer, being reachable directly over the network, so the internal skills were the first to work end to end.

IX. Discussion

Two properties matter more than raw capability in this setting. The first is auditability. Every answer names its sources, the model is forbidden to invent coordinates or legal text, and skills that expose uncertain data say so: the hydrant answer in Figure 1 explicitly flags that OpenStreetMap coverage is uneven and must be cross-checked against the official departmental database. An assistant that is confidently wrong about the position of a water supply is worse than no assistant. Provenance is therefore attached to every answer, including the ones the model is confident about.

The second is fail-closed access control. Professional profiles in a fire and rescue service differ sharply in what they are entitled to consult, and the incident described in Section VIII showed that a language model asked to police its own boundaries will not do so reliably. Enforcement therefore has to be deterministic and to sit outside the model, with the prompt used only as a last layer.

The scope of the deployment is narrower than the architecture makes it sound, and it is worth being plain about that. Only one departmental doctrine corpus is available locally, the search engine being generic over any number of them; several skills depend on datasets that are not yet synchronised; and the model-agnostic service currently runs against providers for which keys exist, with the others visible but disabled. We checked that portability two ways. First, a real cross-provider comparison: the same ten questions, retrieval and prompt, run once each through GPT-4o-mini via a raw API call, exactly as Table III measures Mistral Large, succeeded on nine of ten (the tenth, the wasp-sting query, is discussed below) at a mean latency of 4.3 s and a mean cost of $0.0007 per query, both markedly lower than Mistral Large's 10.3 s and $0.0117, which is the point of a model-agnostic design: the cost-latency-quality trade-off becomes a configuration choice rather than a rewrite. Second, a functional-only check under Claude, a third model family, using the `claude` CLI rather than a raw API call, which returned grounded, sourced answers on all ten questions but cannot be compared on cost or latency, since it necessarily executes through a heavier, general-purpose harness than the raw API calls used for the other two providers. The one GPT-4o-mini failure is informative: given the same correctly expanded retrieval that let both Mistral Large and Claude reason from a wasp sting to the anaphylaxis protocol, GPT-4o-mini refused outright, evidence that the retrieval-quality finding of Section VI does not remove every model-capability gap, only the ones caused by retrieval. The evaluation we report is likewise partial: the doctrine benchmark covers ten questions, each run three times (thirty calls, all successful; 95% Wilson interval treating calls as independent: 88.6–100%), too few distinct topics to support the reported success rate as more than a pilot result, the repetitions bounding run-to-run variance rather than topic coverage, and measures latency, cost and turn count, not factual fidelity, which so far has been verified only by manual inspection on a handful of topics, including verbatim adult and paediatric dosages. A full accuracy audit over a larger panel of questions, and a like-for-like latency and token comparison of the two architectures across all domains rather than doctrine alone, remain to be done.

These, together with the transport-isolating fourth configuration of Section VI-C, a benchmark of the LangChain wrapper itself, rather than the raw API calls reported above, run under at least two providers, and an authorization red-teaming campaign covering allowed, forbidden and mixed requests across roles with adversarial paraphrasing, are revisions that call for a new measurement campaign rather than a rewording of this manuscript. We plan to run them for the extended journal submission that IEEE-SmartData offers to accepted authors, rather than hold this report past the camera-ready deadline for results that are not yet ready.

X. Conclusion

We have described PredictOps, an assistant for the French departmental fire and rescue services, and the two architectural

changes that shaped it. The first generation combined MCP servers with a cloud multi-agent workflow that classified each question and routed it through a tree of specialised agents; it worked, but it spent several sequential LLM turns and several network hops on every request, however trivial the underlying lookup. We replaced it with a pure-skill design in which the assistant runs a self-contained local script and reads its structured output, and we reported what that bought us on the doctrine domain: a single LLM turn per query instead of eight to sixteen, an average of 10.3 s and $0.0117 per query, and a corpus search that runs offline. We then rebuilt the orchestration layer on LangChain so that the same eighteen skills, organised into eight families and covering weather, water, terrain, risk, doctrine, law, health and internal platform data, execute under any supported provider and behind fail-closed role-based access control. Along the way we documented the obstacles that mattered in practice, most of which concerned retrieval quality and silent cross-platform failures rather than model behaviour, and we found repeatedly that generic fixes outperformed narrowly targeted ones.

Several directions follow: a like-for-like comparison of the two architectures across every domain rather than doctrine alone, an accuracy audit over a panel of validated questions since our current benchmark covers cost and latency but not fidelity, synchronising the remaining doctrine corpora and territorial datasets, enabling the model providers currently listed but disabled, aligning authentication with the production identity provider, moving the service from a local installation to production infrastructure, and evaluating the assistant with operational staff in the field, since latency alone cannot tell us whether a departure sheet produced in seconds changes what an officer actually does. The most substantial of the revisions requested by reviewers, a controlled ablation isolating transport, orchestration and retrieval (Section VI-C), a cross-provider benchmark of the LangChain wrapper itself, and an authorization red-teaming campaign, are deferred to the extended journal version discussed in Section IX.

Acknowledgment

This work was supported by a research grant from the Agence Régionale de Santé (ARS) Bourgogne-Franche-Comté. The authors thank the partner departmental fire and rescue services for their operational guidance.

References

[1] Anthropic, "Model Context Protocol specification," 2024. [Online]. Available: https://modelcontextprotocol.io

[2] LangChain, Inc., "LangChain and LangGraph: a framework for building agentic applications," 2024. [Online]. Available: https://www.langchain.com

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. 9459–9474.

[4] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[5] C. E. Van Wagner, "Development and structure of the Canadian forest fire weather index system," Canadian Forestry Service, Forestry Technical Report 35, Ottawa, 1987.

[6] Isara Tech, "THOT: sovereign AI for French firefighters," 2024. [Online]. Available: http://gothot.fr/ (accessed 2026-08-03)

[7] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: synergizing reasoning and acting in language models," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

[8] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: large language model connected with massive APIs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[10] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: solving AI tasks with ChatGPT and its friends in Hugging Face," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[11] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, et al., "AutoGen: enabling next-gen LLM applications via multi-agent conversation," arXiv:2308.08155, 2023.

[12] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, et al., "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, 186345, 2024.

[13] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, et al., "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[14] T. Daniel, A. de Chevigny, A. Champrigaud, J. Valette, M. Sitbon, M. Jardin, D. Chevalier, and S. Renet, "Answering hospital caregivers' questions at any time: proof-of-concept study of an artificial intelligence-based chatbot in a French hospital," *JMIR Human Factors*, vol. 9, no. 4, e39102, 2022.

[15] J. San-Miguel-Ayanz, E. Schulte, G. Schmuck, A. Camia, P. Strobl, G. Liberta, et al., "Comprehensive monitoring of wildfires in Europe: the European Forest Fire Information System (EFFIS)," in *Approaches to Managing Disaster*, InTech, 2012, pp. 87–108.

[16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.