An LLM-Based Multi-Agent Framework for Natural Language Interrogation and Exploitation of Smart City Services

Luca Capece[1], Paolo Nesi[1], **Gianni Pantaleo[1]**

[1]DISIT Lab, Department of Information Engineering (DINFO), University of Florence, Florence, Italy

https://www.disit.org https://www.snap4city.org

NGEN-AI 2026 · The 2026 International Conference on Next Generation AI Systems

1–4 September 2026 · Trento, Italy · https://ngen-ai.org

Motivation: Smart cities are rich in data, but hard to exploit

- Modern **Smart Cities** expose thousands of heterogeneous **data**, **IoT/IoE devices**, **APIs**, **microservices and applications** across many different domains: mobility, environment, energy…
- Urban **Digital Twins** combine data acquisition and analysis, simulation, visualization and decision-support in multilayer architectures.
- High availability of data, APIs and tools **does not imply** that end users, city operators and developers can actually and efficiently exploit them.
- The main limitations, critical aspects and barriers reported in literature are related to **interoperability**, **data integration**, **standardization and usability**:
  - *Narrow tool inventories*
  - *Limited tool interoperability and standardization*
  - *Lack of stateful, multi-step planning*
  - *Limited usability for non-technically skilled users*
  - *Lack of integration into a real multi-domain city platform*

Motivation: Smart cities are rich in data, but hard to exploit

Heterogeneity

Each microservice has its own endpoint, parameters and response format

Fragmentation

Composition across different domains must be often done manually, service by service

Accessibility

Exploitation of applications, APIs and visualization tools often requires technical skills that are not always owned by citizens / final users

- **The gap is not lack of data and/or services: it is their efficient access, integration and exploitation.**
- Problem: How can a non-expert ask, for example, "***Give me a route between two locations and find relevant city services along it***", without knowing which endpoints, schemas and parameters to call?

Contributions of this work

A modular multi-agent framework based on LLM and orchestrated tool calling that turns natural language queries into controlled, multi-step interactions with live urban services.

Model Context Protocol + Tools

MCP connects the LLM to external tools and data through a standardized interface, enabling reliable tool use in agentic workflows.

LangGraph

Design of stateful agent workflows with steps, tools, loops, memory state and conditional decisions.

Cross-domain NL querying
General-purpose access to city services, not a single vertical

Automated orchestration
Orchestration of several smart city Microservices

Standardized tool layer
Microservices, Applications (e.g. Node-RED flows) and REST APIs re-exposed as standard MCP tools

Snap4City integration
Operational integration inside a multi-tenant, multi-user smart-city platform

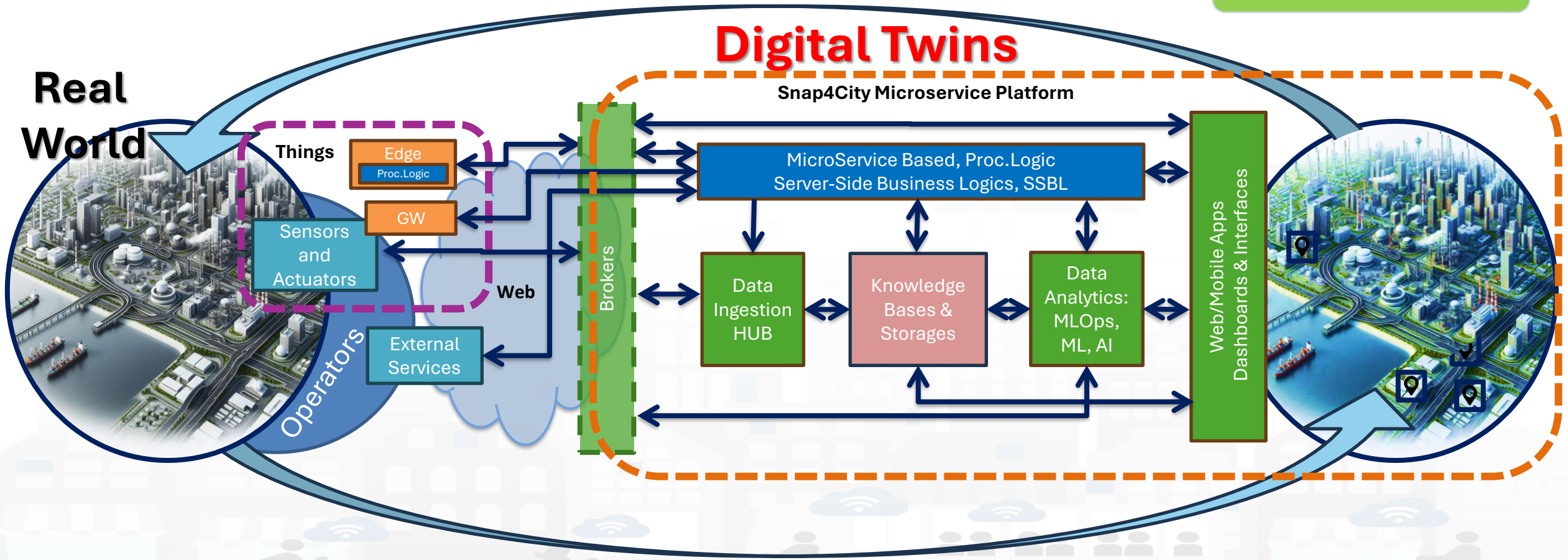
Live services + knowledge graph
Agents interrogate both structured Knowledge Bases and unstructured sources

Composite multi-tool queries
Multi-hop reasoning across domains, with supervision and recovery

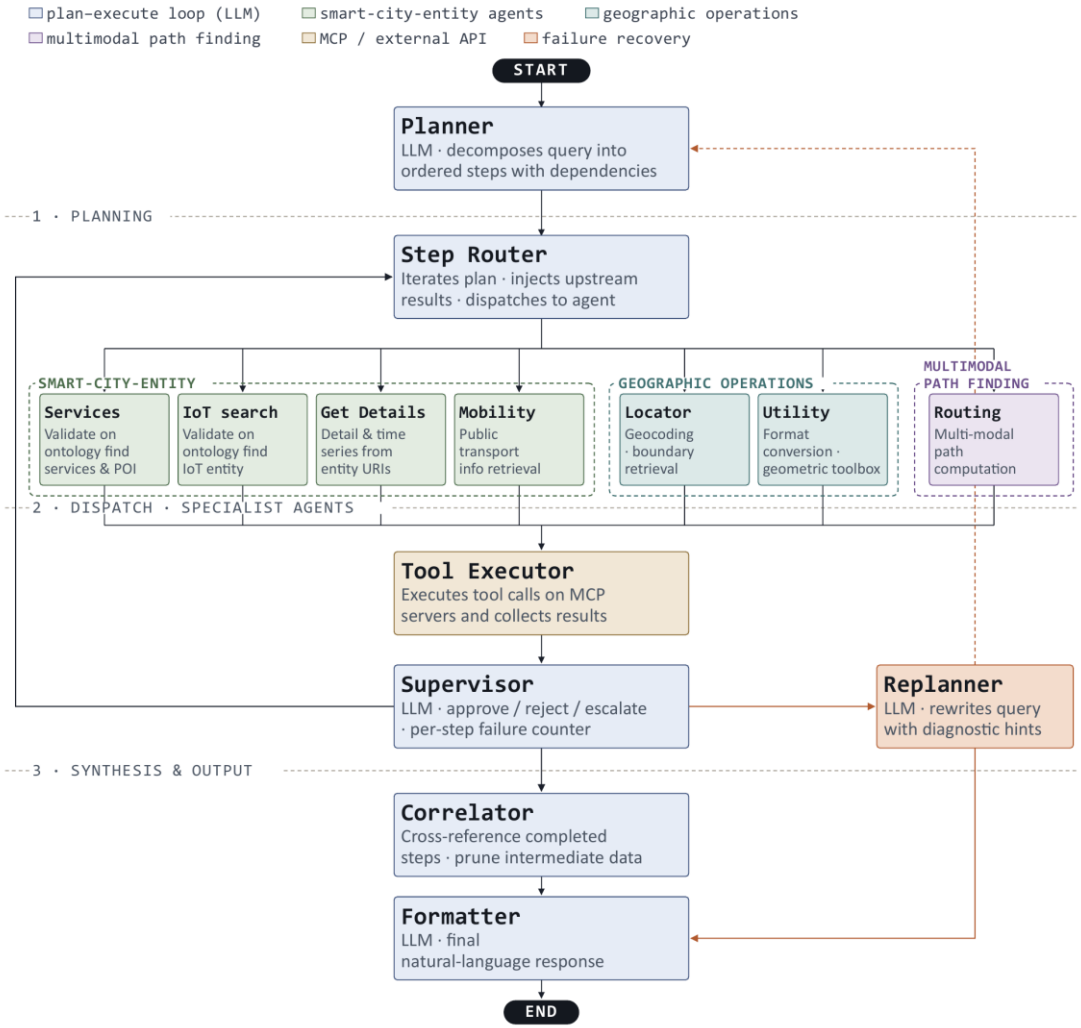
Snap4City Digital Twin Platform

Km4City KG scale

> 1B triples



End-to-end architecture: three layers



1 · PLANNING LAYER

The query is decomposed by the Planner Model into ordered atomic steps with explicit inter-step dependencies; only the declared upstream results are injected into each step.

2 · DISPATCH LAYER

Every step is assigned to a domain-specific agent by the Step Router; MCP tool calls are executed by the Tool Executor and validated by an LLM Supervisor before the plan advances.

3 · SYNTHESIS & OUTPUT LAYER

Completed steps are cross-referenced, unused intermediate data is pruned, and the final natural-language answer is generated.

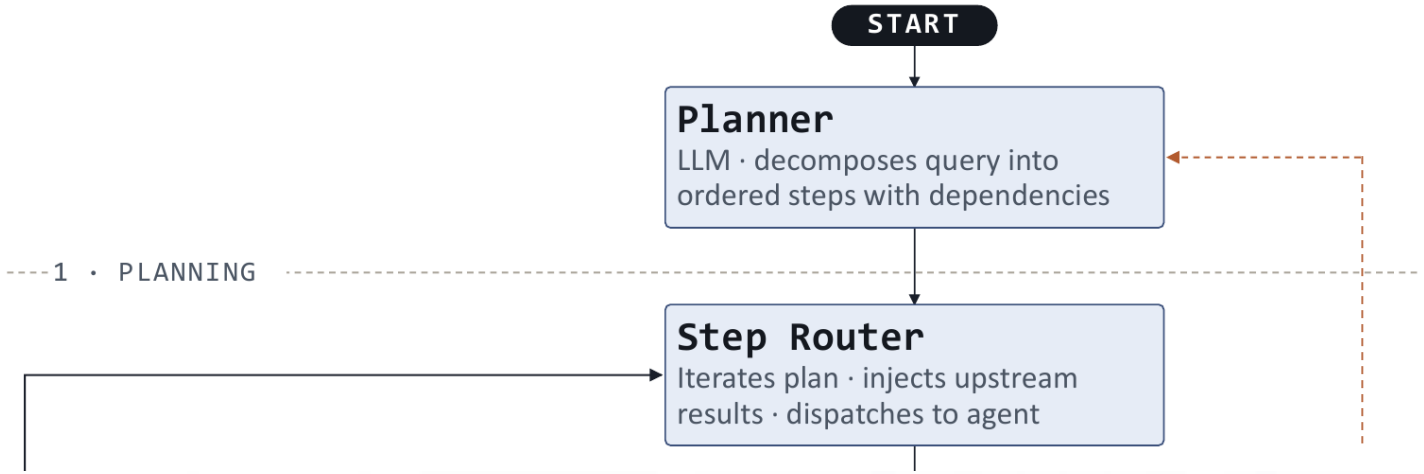
From city microservices to MCP tools

- APIs and microservice documentation has been restructured into **standard**, **typed MCP tool definitions** with machine-readable input schemas.
- **Uniform response structure** (result payload + descriptive error).
- Implicit parameter semantics made explicit:
  - optional vs required fields;
  - enumerated values;
  - coordinate systems;
  - units
- Always updated: Services and IoT agents run **SPARQL queries** (periodically or at run time by configuration) to fetch the current service taxonomy, instead of using a static list.
- Tools grouped by **semantic tags**: compact planning prompts; each agent only sees its own tool subset.

Agents	Tools	Capabilities
Locator	9	Geocoding, administrative boundary retrieval
Services	9	Discovery of services and POIs (spatial or textual)
IoT	4	Sensors and devices by proximity, category, hardware
Get Details	6	Entity metadata, live and historical IoT readings
Mobility	15	Transit networks: agency, line, route, stop, schedules
Routing	4	Multi-modal paths (car, foot, public transport)
Utility	18	Geographic format conversion and geospatial toolbox

65 MCP tools in 7 functional categories — a new domain is added by registering tools and one agent.

Planner and Step Router



- The Planner receives, as part of its instruction, a **description of each specialized agent capabilities**, together with a **graph describing agent dependencies**.
- The Planner reasons over **agent-level capability descriptions and a dependency graph**, not over single tool signatures.
- The planning prompt remains compact as the catalogue grows; only the assigned specialist reads the detailed MCP tool schemas.

Plan-then-execute

The query is decomposed once into an ordered plan of atomic steps, instead of interleaving reasoning and action at every turn.

Self-contained steps

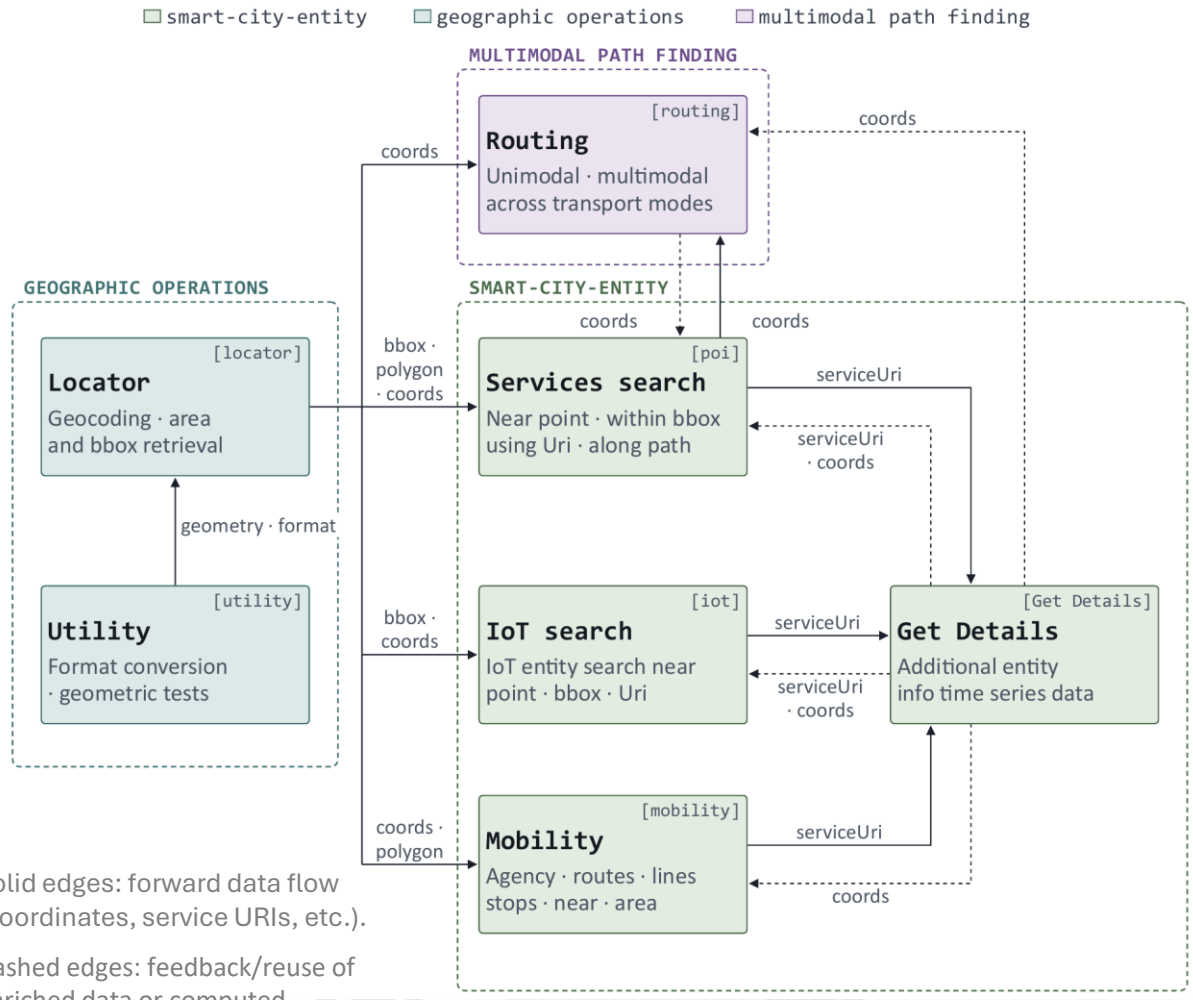
Each step is annotated with:

i. the specialist agent it is assigned to:
ii. a self-contained instruction;
iii. the list of prior steps with the required results.

Context budget

The Step Router injects into each specialist only its task and the upstream results required by its dependencies (***specialist agents never see the full shared state or execution history***).

Specialist agents and dependency graph



- Solid edges: forward data flow (coordinates, service URIs, etc.).
- Dashed edges: feedback/reuse of enriched data or computed geometries.

Seven specialized agents:

- **Locator** — map geographical names to coordinates or administrative boundaries
- **Services / IoT search** — Retrieve POIs and devices in an area, after category validation on the Snap4City taxonomy
- **Mobility** — public transport agencies, lines, routes, bus stops
- **Get Details** — metadata and time series from entity URIs
- **Routing** — multi-modal paths, optionally returning geometry
- **Utility** — format conversion and geometric operations

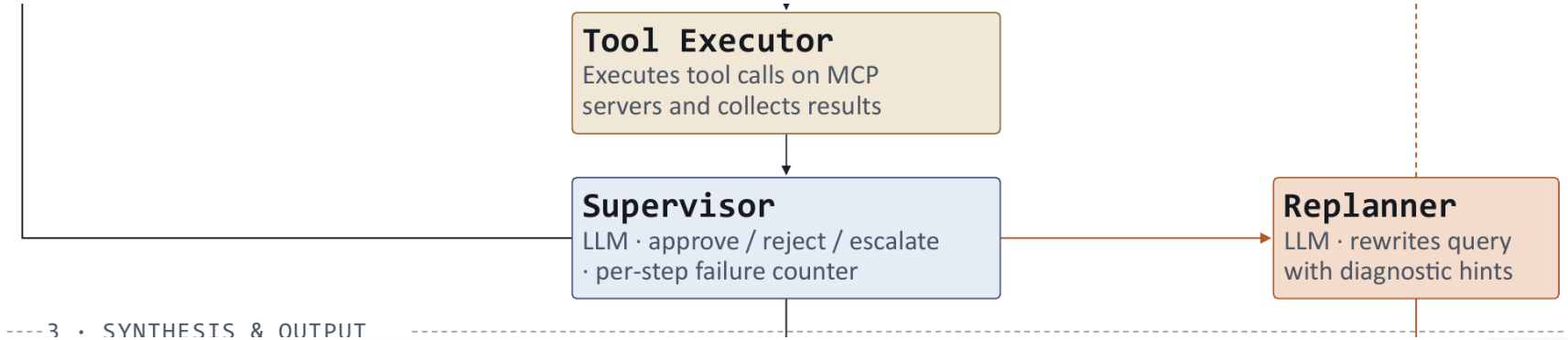
Specialist agents and dependency graph

- Each agent receives its filtered tool set and a prompt composed of a domain-specific instruction, the step's task description, and any injected upstream results.

- Agents exchange data exclusively through the shared state. For any user query, ***each agent only receives a subset of the shared state and the upstream results related to its own declared dependencies***.

- The agent's language model then selects and parameterizes one or more tool calls, which are forwarded to the Tool Executor that invokes the corresponding MCP tools.

Supervision, retry and replanning



Tool Executor:

- Runs the specialist agent's tool calls on the MCP servers and collects the raw results returned by each microservice.
- In order to optimize the usage of the LLM context window, large WKT/GeoJSON payloads are replaced by pointer tokens, which are later dereferenced by the Formatter (full payloads are kept in a separate lookup map).

Supervisor:

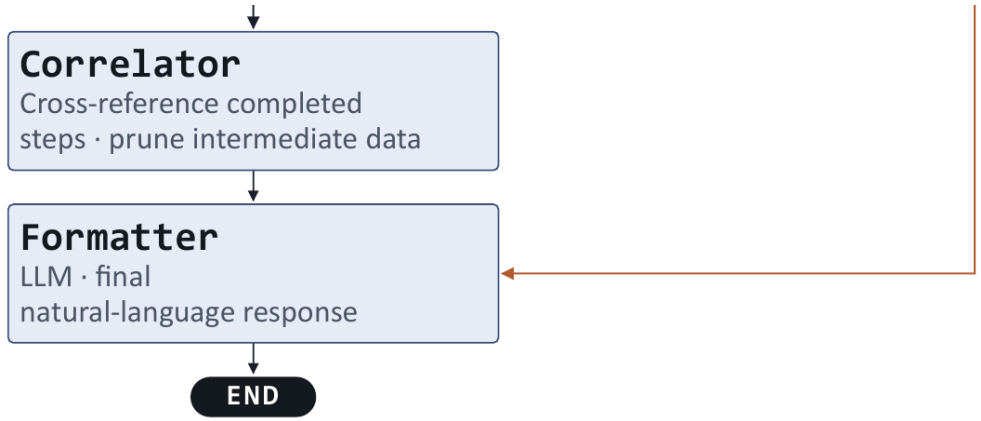
Approve
The Supervisor validates the tool output against the step instruction and advances.

Reject & retry
The step is re-executed with the failure reason appended to the agent context, within a bounded budget.

Escalate & replan
The Replanner rewrites the query preserving intent, adds diagnostic hints and sends it back to the Planner.

Graceful failure
If the replanning budget is exhausted, partial results are kept and the answer states what could not be resolved.

Synthesis: correlation and formatting



The answer is grounded exclusively in data that was actually retrieved and cross-referenced.

Correlator

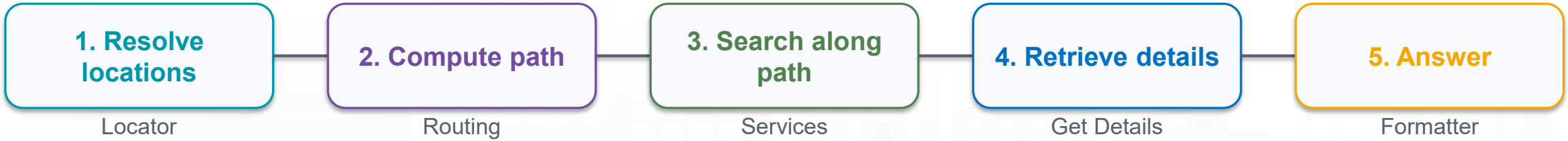
- Uses the dependency annotations to cross-reference results across steps, by matching parent output values against the arguments used in the child's tool calls
- Prunes intermediate records that no downstream step has consumed
- Additional consistency check: results approved by the Supervisor but not consumed by any downstream step are flagged, not hidden

Formatter

- Receives the correlated report plus the original request and generates the final natural-language answer
- Pointer tokens are resolved back to their original values before the response is returned

Running example: a composite urban request

"Give me the driving route from point A to point B and list the restaurants along the route."



Separation between reasoning and execution

- Two concerns usually conflated in monolithic conversational agents are **kept separate**: natural-language reasoning and domain-specific tool execution
- The **tool layer** handles the platform complexity: endpoints, formats, access patterns
- The **orchestration layer** handles planning, dependency tracking and failure recovery with no knowledge of the invoked services
- Reasoning and execution may connect only through two interfaces: the **plan** and the **shared state**

Adding a new city-service domain

- Register the new MCP tools
- Tag them with a new category
- Add one specialist agent to the dependency graph

No existing agent or orchestration node is modified.

Ready for many other Snap4City services:

Visual analytic · What-if scenarios · Origin-Destination matrices · ML/AI predictions, anomaly detection, optimization, simulation

Validation: dataset and evaluation axes

Evaluation dataset

- **Structured:** 30 queries on the Snap4City knowledge base, each paired with an equivalent hand-written SPARQL query as ground truth
- **Unstructured:** 90 smart-city queries — 35 written directly by domain experts, 55 generated with LLM assistance and reviewed by them

30
SPARQL-backed queries

90
expert-rated queries

4
evaluation axes

Four complementary axes

(i) Agent selection reliability
Does the Planner route each atomic step to the right specialist?

(ii) Retrieval accuracy
Agent answers compared to the reference SPARQL result sets on the KG.

(iii) Human evaluation
Domain experts rate freely formulated queries on a 1–5 scale.

(iv) Answer quality
G-Eval Correctness, agentic system vs. tool-free ablation, three judges.

Agent selection accuracy

Agent	Precision	Recall	F1-score
Locator	92.6%	98.0%	95.2%
Services	92.3%	100%	96.0%
Get Details	69.6%	80.0%	74.4%
IoT	100%	76.9%	87.0%
Mobility	90.0%	100%	94.7%
Routing	88.9%	88.9%	88.9%
Overall	89.9%	93.6%	90.5%

The Utility agent supports the other specialists rather than retrieving data, so it is not evaluated here.

117 of 126 dispatches matched the expert annotation.

Get Details — lowest precision (0.696)

The framework tends to explore unstructured data in depth before answering, generating extra planning steps.

IoT — lowest recall (0.769)

Sensor-related intents are absorbed by more generic agents: in a smart city a "parking" is both a physical lot and the sensor reporting free slots.

Retrieval accuracy against SPARQL ground truth

0.929
Precision (per-query mean)

0.623
Recall (per-query mean)

0.700
F1-score (per-query mean)

- **High precision:** almost every item returned is genuinely part of the ground-truth answer — hallucinated or spurious results are rare
- **Lower recall:** the shortfall is concentrated in **high-cardinality queries**, where tools paginate or cap records and the limited context window forces the LLM to return a subset
- Consistent with the design priority of the framework: **precise, verifiable answers over exhaustive enumeration**

Human evaluation by domain experts

Protocol

- 90 freely formulated smart-city queries, each answer scored from 1 to 5 by domain experts
- **True positive**: an attempted answer rated at least 3 stars
- **False positive**: an attempted answer rated less than 3 stars
- **False negative**: a query the system should have resolved but did not
- Plans averaged roughly 4 to 8 atomic operations (substantially harder than single-hop question answering)

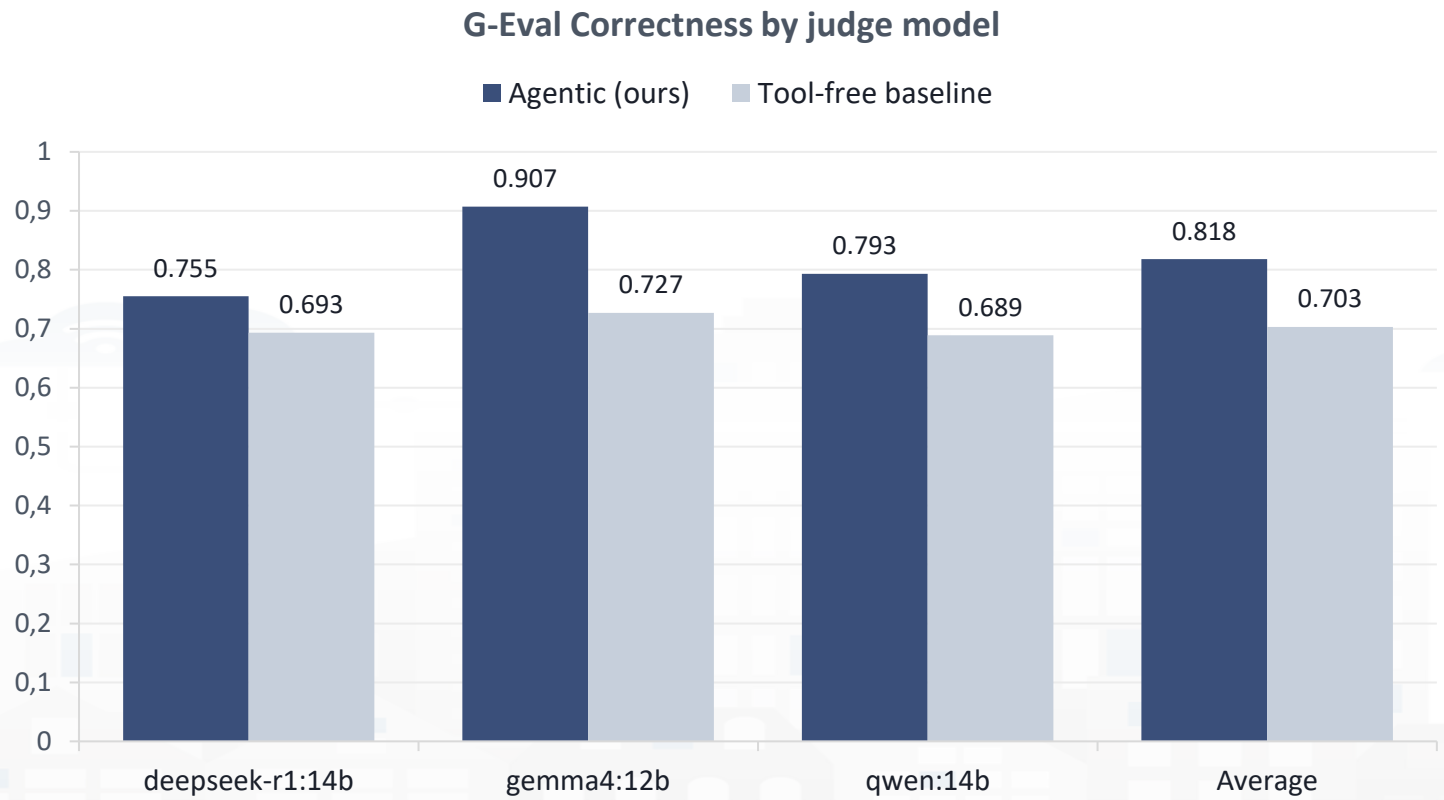
0.829	0.907	0.866
Precision	Recall	F1-score

Results discussion:

High recall: the system attempts and resolves the large majority of requests.

The slightly lower precision reflects answers judged incomplete or partially off-topic (most of them are the same high-cardinality cases seen in the SPARQL retrieval). This is a target for future refinement.

Answer quality: agentic system vs. tool-free ablation



Correctness and relevance plus numerical accuracy, scored with DeepEval on log-probability weighted judgements; three judges are used to mitigate LLM-evaluator bias.

+0.115 on average G-Eval score …

Grounding the answer in tool-retrieved data measurably improves factual accuracy on every judge model.

… at a higher cost

≈67 s per agentic query vs 6 s for the baseline · ≈12 tool calls per query · a replan triggered in 36% of runs.

Conclusions and future work

- A **general-purpose, extensible and operational** agentic framework for natural-language interrogation of smart-city services, integrated in the Snap4City platform
- Planning, specialist agents, MCP tool definitions, supervision, recovery and correlation turn complex intents into **controlled multi-step interactions with live urban services**
- The Planner routes atomic steps to the right specialist with **high precision**; structured retrieval against the KG achieves **high precision**;
- The agentic architecture **improves answer correctness over standalone text generation** on all three judge models
- **Limitations**: Recall is limited for high-cardinality result sets; Coverage of a subset of Snap4City tools

Future work

- Extend the tool catalogue and the set of specialist agents and MCP tools. To this purpose, Snap4City already has in place services such as data analytics, optimization, simulation, ML/DL predictive models, RAG
- Enlarge the use cases beyond the validated datasets

Thank you

Questions are welcome