

Progetto GAINS

(Generative AI for Network Sustainability)

Deliverable D3.2

Technical Report su Progettazione e Implementazione Network Orchestrator

Giordano, A. - Basile, L. - De Giacomo, A. - Haider, B. - Paesani, M.
2026-01

Work Package: WP3 Progettazione, Implementazione Prototipale e Valutazione degli Agenti

Stato: Definitivo | Versione: 1.0

Classificazione: uso interno di progetto; condivisione controllata con partner su necessità.

Abstract

La gestione operativa delle reti e delle infrastrutture digitali di nuova generazione richiede processi di change sempre più rapidi, controllabili e auditabili, a fronte di una complessità crescente dovuta a eterogeneità tecnologica, dinamiche di traffico variabili, vincoli di sicurezza e requisiti di sostenibilità. In questo contesto, l'automazione tradizionale basata su playbook e regole statiche mostra limiti strutturali: fatica a generalizzare tra domini e vendor, non esplicita in modo robusto le assunzioni decisionali, e raramente integra in modo nativo un modello di accountability che colleghi decisione, evidenze, configurazioni e impatto. Questo deliverable presenta la progettazione e l'implementazione di riferimento del Network Orchestrator, un agente software orientato a intent-based networking e governance del change, progettato per trasformare intent operativi in configurazioni candidate e piani di esecuzione, integrando knowledge grounding e stima dell'impatto energetico in modo tracciabile e riproducibile. L'approccio proposto combina componenti deterministiche di controllo con componenti assistive basate su modelli linguistici, vincolate da schemi e regole di validazione. Il Network Orchestrator opera come servizio API stateless con persistenza esterna di stato e artefatti, introducendo un lifecycle del job esplicito e idempotente, capace di gestire end-to-end la catena submit-context-generation-validation-impact-documentation-approval. Il cuore architetturale è una pipeline agentica che: normalizza l'intent in una rappresentazione canonica; effettua retrieval applicabile da un knowledge store mediante filtri hard su vendor, ruolo e versione; produce artefatti strutturati che includono riferimenti espliciti alle fonti usate; applica policy deterministiche di sicurezza e conformità; esegue validazioni sintattiche e semantiche su configurazione e change plan; stima l'impatto energetico as-is/to-be/delta tramite telemetria e modelli, esplicitando confidenza, metodo e assunzioni; infine genera un explain pack operativo orientato a revisione, audit e decisione umana. L'uso dell'endpoint LLM esterno OpenAI-compatibile è confinato a task non-authoritative come parsing assistito, binding di variabili e arricchimento controllato del piano, con minimizzazione del contesto, redazione preventiva e post-validazione a schema, mitigando rischi di prompt injection e data leakage. Il

contributo principale è un modello implementativo replicabile che rende la generazione di change “decisionabile”: ogni output è pacchettizzato in artefatti versionati e hashati, collegati a knowledge references ed evidence references, e accompagnati da report di validazione e da condizioni di rollback verificabili. Questa impostazione abilita governance del change coerente con requisiti industriali di tracciabilità, riduzione del rischio operativo e controllo di conformità, e introduce un primo livello di sostenibilità by design grazie alla produzione sistematica di stime d’impatto e alla gestione esplicita dell’incertezza. La valutazione è definita tramite una suite multilivello di test unitari, contract test degli schemi, test di integrazione e regressione, includendo metriche quantitative su correttezza, sicurezza, auditabilità e prestazioni, oltre a test avversari per robustezza. Il Network Orchestrator costituisce un tassello chiave per l’industrializzazione di architetture agentiche applicate a reti e infrastrutture, in linea con gli obiettivi di transizione digitale e sostenibile: riduce il time-to-change, aumenta la qualità e la verificabilità delle decisioni, e prepara l’integrazione controllata verso sistemi di esecuzione e closed-loop monitoring, mantenendo al centro riproducibilità, accountability e protezione dei dati. [15][22][23][27]

Indice

1. Scopo e perimetro del Network Orchestrator
 - 1.1 Obiettivi funzionali
 - 1.2 Assunzioni e vincoli operativi
 - 1.3 Interfacce con gli altri componenti di piattaforma
2. Requisiti e criteri di progetto
 - 2.1 Requisiti funzionali
 - 2.2 Requisiti non funzionali sicurezza, audit, resilienza, qualità
 - 2.3 Criteri di replicabilità industriale e portabilità
 - 2.4 Principi di sostenibilità e footprint attribuibile
3. Architettura logica del Network Orchestrator
 - 3.1 Componenti interni Intent Normalizer, Planner, Validator, Artifact Manager
 - 3.2 Flussi principali intent, plan, approval, artifacting, degrade
 - 3.3 Integrazione con knowledge e telemetria
 - 3.4 Confini di responsabilità e threat model
4. Modello dati e schemi canonici

- 4.1 Oggetti core OrchestrationJob, CandidateConfigurationArtifact, ChangePlanArtifact, FootprintEstimateArtifact
- 4.2 Versioning, hash, immutabilità e lineage
- 4.3 Riferimenti knowledge refs ed evidence refs
- 4.4 Esempi completi di payload JSON
- 5. Contratti API del Network Orchestrator
 - 5.1 Endpoint pubblici e autenticazione
 - 5.2 Error model, idempotenza e correlazione richieste
 - 5.3 Job lifecycle stati, transizioni, policy di degradazione
 - 5.4 Esempi di chiamate e risposte
- 6. Pipeline agentica e logiche di generazione
 - 6.1 Intent parsing e normalizzazione nel modello canonico
 - 6.2 Retrieval applicabile e grounding su blueprint, snippet e policy
 - 6.3 Generazione candidate configuration e change plan
 - 6.4 Validazioni statiche e controlli di consistenza
 - 6.5 Stima impatto e sostenibilità as is, to be, delta, confidenza
 - 6.6 Human in the loop approve reject e audit trail
 - 6.7 Output finali, explain pack e condizioni di rollback
- 7. Implementazione software
 - 7.1 Struttura repository e moduli
 - 7.2 Configurazione runtime env, secrets, profiles
 - 7.3 Prompting strategy controllata e template
 - 7.4 Driver di validazione e render per vendor, device_role, OS
 - 7.5 Logging, tracing e audit trail
- 8. Deployment in Docker
 - 8.1 Prerequisiti e profilo di esecuzione
 - 8.2 docker compose completo servizi, volumi, reti
 - 8.3 Bootstrapping inizializzazione schemi e smoke test
 - 8.4 Hardening minimo per ambienti enterprise
- 9. Testing e valutazione

9.1 Unit test e contract test

9.2 Test di integrazione con knowledge e telemetria

9.3 Test di robustezza regressione sicurezza

9.4 Metriche di qualità correttezza, applicabilità, riproducibilità, tempi

10. Limitazioni, rischi e roadmap evolutiva

10.1 Failure mode e mitigazioni

10.2 Evoluzioni previste e generalizzazione industriale

Bibliografia

Appendici

A. Esempi end to end intent plan approval output

B. Dizionario campi e convenzioni

C. Checklist operativa per deployment e verifica

D. Acronimi essenziali