UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica

Project Documentation

2024/2025

course: Autonomous Systems

Network Managment System

Team Members		
Name & Surname	**Matriculation Number**	**E-mail address**
Agostino D'Agostino	*303226*	*Agostino.DAgostino@student.univaq.it*
Alessandro DiGiacomo		*Alessandro.DiGiacomo@student.univaq.it*

Repository:

1. Functional Requirements

Requisito Funzionale	Priorità	Motivazione
Rilevamento metriche di rete da sensori OSGi	Alta	Fondamentale per il monitoraggio della rete in tempo reale
Analisi automatica di anomalie tramite regole o ML	Alta	Permette risposta tempestiva a condizioni anomale
Generazione di piani d’azione tramite motore di regole	Alta	Componente centrale del comportamento autonomo del sistema
Esecuzione automatica delle azioni tramite attuatori	Alta	Realizza modifiche operative senza intervento umano
Notifiche e aggiornamenti in tempo reale	Media	Migliora l’esperienza utente e la trasparenza del sistema

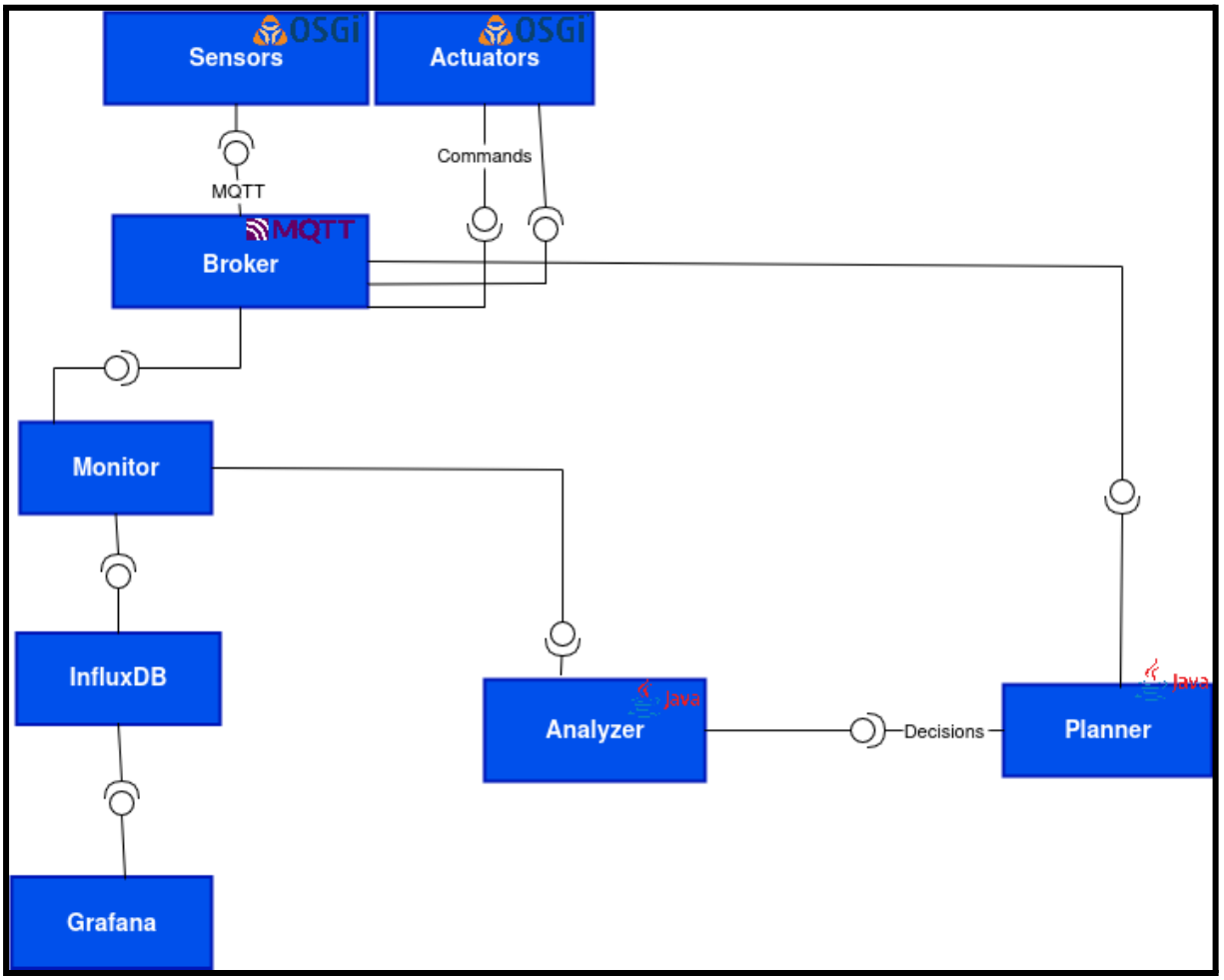
2. Non-Functional Requirements

Requisito Non Funzionale	Priorità	Motivazione

Manutenibilità e modularità dell'architettura	Alta	Essenziale per aggiornamenti futuri e integrazione di nuovi componenti
Tempo di risposta inferiore a 3 secondi	Media	Migliora la reattività e usabilità del sistema
Scalabilità tramite deployment multi-container		

Architettura

Component Diagram



Componenti Principali

Il sistema è progettato secondo un'architettura a microservizi basata su principi event-driven, con una chiara separazione dei compiti tra i componenti del ciclo MAPE-K. L'infrastruttura è completamente containerizzata tramite Docker e orchestrata mediante Docker Compose.

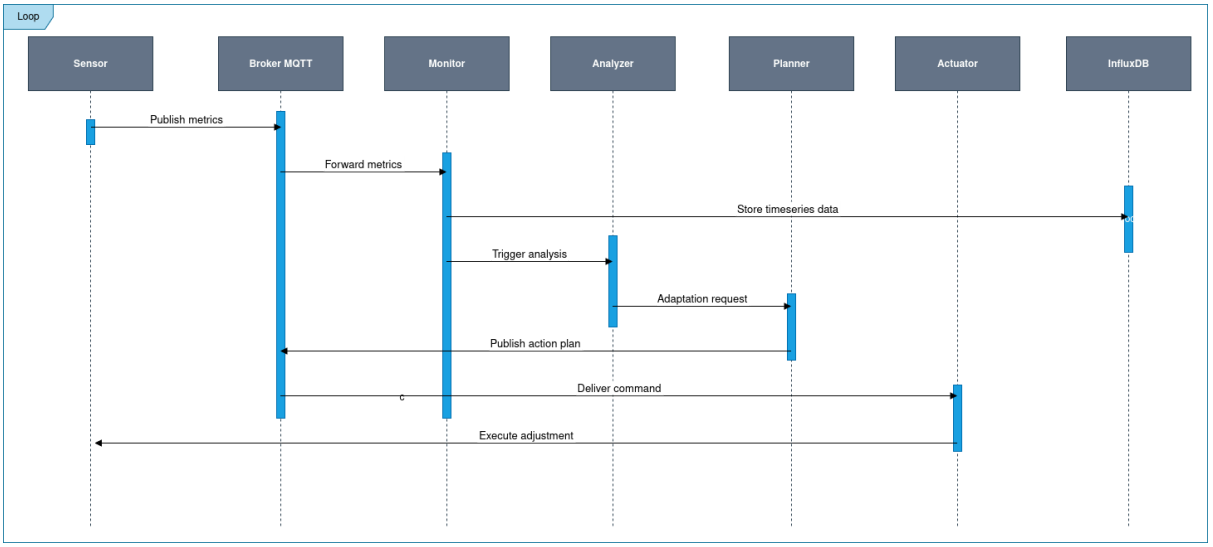
Il **MQTT Broker** (Eclipse Mosquitto) funge da sistema nervoso centrale per la comunicazione asincrona, implementando pattern pub/sub per connettere tutti i componenti. I **Sensori OSGi**, implementati come bundle dinamici Java, raccolgono metriche di rete in tempo reale pubblicando dati sul broker con QoS 1 per garantire delivery affidabile.

Il servizio **Monitor** (Python) opera come gatekeeper dei dati: si sottoscrive ai flussi telemetrici, esegue validazione e arricchimento dei payload, e persiste le metriche su **InfluxDB** tramite scritture asincrone batch-ottimizzate. L'**Analyzer** implementa la logica di anomaly detection combinando regole statiche (soglie predefinite) con modelli ML, pubblicando decisioni di adattamento su topic dedicati.

Il **Planner** costituisce il cervello decisionale: elabora gli input dell'Analyzer generando piani d'azione attraverso un motore rule-based (Drools), con politiche di priorità differenziate per scenari critici. Gli **Attuatori OSGi** traducono i comandi ricevuti in azioni concrete sulla rete fisica, completando il ciclo di controllo.

Grafana fornisce il layer di visualizzazione, collegandosi a InfluxDB tramite connettori nativi per dashboard real-time.

Sequence Diagram



9. Used Technologies

Tecnologia	Ruolo
Docker / Docker Compose	Containerizzazione e orchestrazione dei servizi
MQTT (Mosquitto)	Protocollo pub/sub per comunicazione asincrona tra componenti
Java + OSGi	Implementazione modulare dei sensori e attuatori
Python	Monitor e data pipeline
InfluxDB	Database per metriche time-series
Grafana	Dashboard di visualizzazione in tempo reale
Jackson, Paho MQTT	Parsing JSON e client MQTT in Java

Principi Applicati

- **Self-Configuration**
  Il sistema è in grado di **configurarsi automaticamente** al momento dell'avvio o durante la riconfigurazione dinamica. I componenti (sensori, attuatori, monitor, planner) vengono registrati e connessi tra loro senza necessità di intervento manuale.

- **Self-Monitoring**
  I sensori OSGi rilevano metriche ambientali e di rete in tempo reale. Il monitor effettua una **raccolta continua** dei dati per mantenere una visione costante dello stato del sistema.

- **Self-Analysis**
  Il modulo Analyzer interpreta i dati raccolti per **identificare condizioni anomale**, sfruttando regole statiche o modelli predittivi. Questa capacità permette al sistema di

comprendere autonomamente se è necessaria un’azione.

- **Self-Planning**
  In caso di anomalie, il Planner elabora **strategie di adattamento** che rispondono a obiettivi di efficienza, affidabilità e resilienza. I piani sono generati dinamicamente tramite un motore di regole (Drools), adattabili al contesto attuale.

- **Self-Adaptation (Execution)**
  Gli attuatori implementano i piani ricevuti dal Planner traducendoli in **azioni concrete** sul sistema. Queste possono riguardare la modifica di parametri, la riassegnazione di risorse o l'invio di comandi di rete.
- **Self-Adaptation (Execution)**
  Il sistema ha un continuo riadattamento dei parametri per mantenere il tutto all’interno dei limiti definiti dai thresholds.

Benefici dell’approccio self-managed

- **Riduzione dei costi operativi** grazie all’automazione dei processi di gestione.

- **Maggiore resilienza** in scenari dinamici o critici della rete.

- **Scalabilità** del sistema senza incremento proporzionale della complessità.

- **Migliore esperienza utente**, grazie a tempi di risposta rapidi e adattamenti invisibili all’utente finale.