Self-Adaptive Smart Parking Lot Manager

Course: Software Engineering for Autonomous Systems
Team Members: Jawayria Hashmi, Lameya Islam

1. Introduction

Traditional parking management requires constant human oversight and manual adjustments. Modern parking systems face challenges such as inefficient space utilization, queue buildup and congestion during peak hours, revenue loss due to rejected vehicles, and limited real-time visibility. An autonomous, self-adaptive system can address these challenges by reacting to changing conditions automatically, optimizing multiple competing objectives and adapting strategies without human intervention.

This project presents a **Self-Adaptive Smart Parking Lot Manager**, an autonomous system that dynamically regulates parking operations using the **MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge)** autonomic control loop. The system continuously monitors parking lot conditions through simulated sensors and autonomously adapts its behavior to optimize utilization, minimize congestion, and maximize revenue without requiring human intervention.

The project simulates a multi-lot parking system with:

- 2 parking lots (Lot 1: 100 spaces, Lot 2: 50 spaces)
- Dynamic pricing ($1-$20 range)
- Automated gate control (open/close based on queue length and occupancy)
- Vehicle redirection between lots
- Real-time monitoring dashboard

2. System Goals and Objectives

2.1. Primary Goals

#	Description	Metric	Target
1	Maintain Optimal Utilization	Occupancy %	70-95%
2	Minimize Congestion	Queue Length	< 8 vehicles
3	Reduce Rejections	Rejected Count	Minimize
4	Maximize Revenue	Total Revenue	Maximize

2.2. Adaptation Objectives

1. **Dynamic Pricing**: Adjust prices based on occupancy levels
   - Increase price when occupancy > 90% (discourage arrivals)
   - Decrease price when occupancy < 50% (attract customers)
2. **Gate Control**: Manage vehicle flow automatically
   - Close gate when queue ≥ 8 vehicles OR occupancy ≥ 98% (prevent overflow)
   - Reopen gate when occupancy drops below 85% AND queue < 8
3. **Vehicle Redirection**: Balance load across lots
   - Redirect vehicles when one lot reaches 98% capacity
   - Direct to alternative lot with lower occupancy

3. Architecture Design

3.1. High Level Architecture

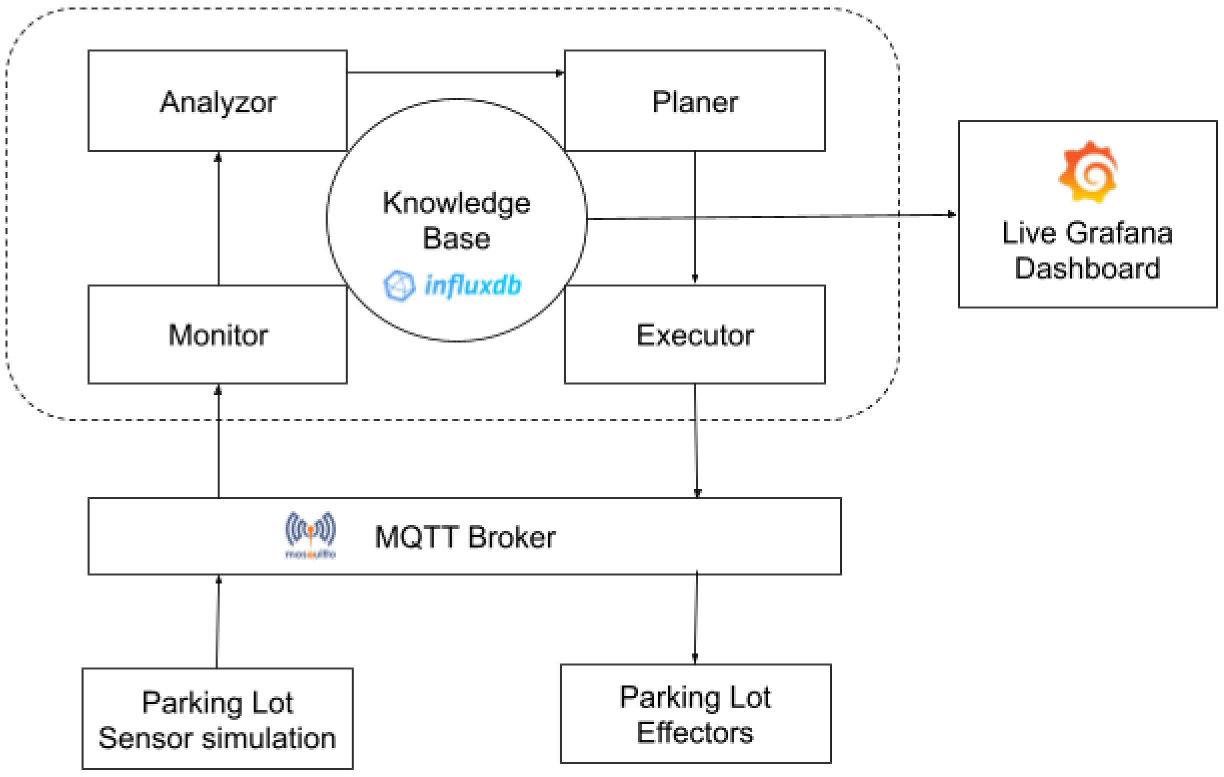

Figure: *High Level Architecture Diagram for Self Adaptive Smart Parking Lot Manager*

3.2. Component Description

Component	Technology	Purpose
Autonomic Manager	Python	MAPE-K control loop implementation

Message Broker	MQTT (Mosquitto)	Asynchronous communication
Knowledge Base	InfluxDB	Persistent storage of system state and history
Dashboard	Grafana	Real-time visualization
Parking Simulator	Python	Simulates parking lot behavior

All services are deployed using Docker Compose to ensure reproducibility and isolation. The system three containerized services:

- MQTT Broker: Eclipse Mosquitto on port 1883
- InfluxDB: Time-series database on port 8086
- Grafana: Visualization dashboard on port 3000

3.3. Communication Architecture

The system uses MQTT publish/subscribe messaging for loose coupling between components. The following topics are used:

Topic Pattern	Direction	Purpose
parking/lot/{lot_id}/sensors	Lot → Manager	Sensor data
parking/lot/{lot_id}/control	Manager → Lot	Control commands
parking/lot/{lot_id}/status	Lot → Manager / Storage	Status updates
parking/system/metrics	Manager → All	System-wide metrics
parking/system/adaptations	Manager → Log	Adaptation events

Message Flow:

1. Parking lot simulators publish sensor data every 2 seconds.
2. Autonomic manager subscribes to sensor data.
3. The manager analyzes data and makes decisions.
4. The manager publishes control commands.
5. Lots receive and execute commands.
6. All data stored in InfluxDB for persistence.
7. Grafana queries InfluxDB for visualization.

4. MAPE-K Loop Implementation

4.1. Monitor Component

Purpose: Collect real-time sensor data from all parking lots

The Monitor component subscribes to MQTT topics for each parking lot and collects the following data at 2-second intervals:

Field	Description
occupancy_percentage	Current lot utilization (0-100%)
current_occupancy	Number of parked vehicles
queue_length	Vehicles waiting at entrance
rejected_count	Cumulative rejected vehicles
current_price	Current parking rate
gate_state	“open” or “closed”
revenue	Cumulative revenue
external_traffic_level	Simulated traffic intensity (0.0-1.0)

Upon receiving sensor data, the Monitor stores it in the Knowledge Base and triggers a MAPE-K analysis cycle.

4.2. Analyze Component

Purpose: Detect deviations from system goals and assess severity

The Analyze component evaluates the current state against predefined thresholds and classifies issues by severity:

Condition	Severity	Issue Detected
Occupancy > 98%	Critical	Capacity overload
Occupancy > 90%	High	High congestion
Queue ≥ 8	High	Entrance congestion
Occupancy < 50%	Medium	Underutilization
Queue ≥ 5	Low	Queue building

The analysis produces an AnalysisResult containing a list of detected issues, severity classification, current system state and recommended adaptation actions.

4.3. Plan Component

Purpose: Select appropriate adaptation actions based on analysis

The Plan component uses a rule-based decision matrix to select actions:

Trigger Condition	Action	Parameters
Occupancy > 90%	INCREASE_PRICE	+$1.00 per step
Occupancy < 50%	DECREASE_PRICE	-$0.50 per step
Queue ≥ 8	CLOSE_GATE	-
Gate closed & Occupancy < 85% & Queue < 8	OPEN_GATE	-
Occupancy > 98% & Queue > 0	REDIRECT_VEHICLES	target_lot

A 10-second cooldown period between adaptations per lot prevents oscillation and ensures system stability. The Plan component creates an AdaptationDecision containing selected actions (limited to 2 per cycle to prevent conflicting or excessive adaptations), trigger conditions, expected outcome prediction and confidence level.

4.4. Execute Component

Purpose: Apply planned adaptations via MQTT commands

The Execute component translates adaptation decisions into control commands and publishes them to the appropriate MQTT topics. Each command includes: - Target lot identifier - Action type (INCREASE_PRICE, CLOSE_GATE, etc.) - Action parameters (new price, target lot for redirection) - Timestamp and reason for audit trail

After execution, the decision is stored in the Knowledge Base for historical analysis and the cooldown timer is reset.

4.5. Knowledge Component

Purpose: Persistent storage of system state, history, and rules

The Knowledge Base uses InfluxDB to store three types of measurements:

Measurement	Purpose	Key Fields
parking_sensor_data	Historical sensor readings	occupancy, queue, price, revenue
adaptation_decisions	Decision audit trail	action, trigger, confidence
system_metrics	Aggregated system state	total_utilization, total_revenue

The Knowledge Base provides query interfaces for retrieving current state for analysis, computing trends and averages and accessing adaptation history for learning. It also supports retrospective analysis through historical queries.

5. Adaptation Strategies

5.1. Dynamic Pricing Strategy

Objective: Balance demand through price elasticity

The dynamic pricing strategy adjusts parking rates based on current occupancy levels:

- **High Occupancy (> 90%)**: Increase price by $1.00 per adaptation cycle to discourage new arrivals
- **Low Occupancy (< 50%)**: Decrease price by $0.50 per adaptation cycle to attract customers
- **Normal Range (50-90%)**: Maintain current price

5.2. Gate Control Strategy

Objective: Prevent queue buildup and system overload

The gate control strategy manages vehicle flow through automatic gate operation:

Close Gate: Queue length ≥ 8 vehicles OR Occupancy ≥ 98%

Open Gate: Occupancy < 85% AND Queue < 8 (both conditions required)

When the gate is closed, new arrivals join the queue (up to maximum 15 vehicles), departures continue normally, reducing occupancy, after which queued vehicles enter as spaces become available.

The dual-condition requirement for gate reopening prevents premature opening when the queue is still high, which was identified as a critical issue during testing.

5.3. Vehicle Redirection Strategy

Objective: Balance load across multiple lots

When a lot reaches critical occupancy (≥ 98%) and has vehicles waiting in the queue, the system attempts to redirect incoming vehicles to an alternative lot with occupancy below 85%.

This strategy reduces rejection rates, improves overall system utilization, balances revenue across lots.

6. Implementation Details

6.1. Key Components

MAPEKAutonomicManager: Central controller implementing the MAPE-K loop with methods for each phase (monitor, analyze, plan, execute) and a main cycle that runs every 2 seconds upon receiving sensor data.

ParkingLotSimulator: Simulates realistic parking lot dynamics including: - Configurable arrival/departure probabilities - Peak hour traffic variation - Queue management with maximum limits - Revenue tracking and gate state management

KnowledgeBase: Manages all InfluxDB interactions for storing sensor data, adaptation decisions, and system metrics. Provides query interfaces for trend analysis and state retrieval.

6.2. Configuration Parameters

Key configurable thresholds (defined in config/config.yaml):

Parameter	Default	Description
target_utilization_min	0.70	Lower bound of optimal utilization
target_utilization_max	0.95	Upper bound of optimal utilization
high_occupancy_threshold	0.90	Trigger for price increase
low_occupancy_threshold	0.50	Trigger for price decrease
critical_occupancy_threshold	0.98	Trigger for redirection
gate_close_queue_threshold	8	Queue length to close gate
gate_reopen_occupancy	0.85	Occupancy to reopen gate
price_increase_step	1.0	Dollar amount per increase
price_decrease_step	0.5	Dollar amount per decrease

6.3. Dashboard Visualizations

The Grafana dashboard provides real-time visibility with four main sections:

1. **System Overview**: Overall utilization gauge, total revenue, combined queue length, rejected vehicles count
2. **Per-Lot Status**: Gate status indicators, individual queue lengths, current price displays
3. **Trend Analysis**: Occupancy time-series, dynamic pricing history, queue length patterns, revenue accumulation
4. **Adaptation Log**: Recent adaptation decisions table with trigger conditions, actions, and confidence levels

7. Conclusion

This project successfully implements a **Self-Adaptive Smart Parking Lot Manager** using the MAPE-K autonomic computing reference architecture. The system demonstrates:

- **Autonomous Operation**: No human intervention required for normal operation
- **Goal-Directed Behavior**: Maintains utilization, minimizes queues, maximizes revenue

- **Reactive Adaptation**: Responds to environmental changes immediately
- **Observability**: Real-time dashboard provides full system visibility

The project also highlights the importance of stability mechanisms such as cooldown periods and multi-condition triggers to avoid oscillatory behavior. Overall, the system demonstrates how autonomic control loops can effectively manage complex, dynamic environments through well-defined goals and feedback-driven adaptation.

8. Limitations and Future Work

The current system is rule-based and does not learn from past adaptations. Future work could extend the Knowledge component with learning mechanisms to adjust thresholds dynamically. Additionally, real-world deployment would require integration with physical sensors and actuators, as well as robustness against sensor noise and communication delays.