SENG 696 Agent-Based Software Engineering

Multi-Agent Daily Planning System (MADPS)

Report 1B
GAIA Design

Ali Mohammadi Ruzbahani [30261140], Shuvam Agarwala [30290444]

Course
Agent-Based Software Engineering (SENG 696)

Instructor: Professor Behrouz Far

Date: October 8, 2025

PubArx:2026.00006v1 [cs.AI] 4 Nov 2026 · Preprint

Contents

1 Methodology & Overview **5**
- 1.1 Design Objectives . . . 5

2 Goals & Traceability **6**
- 2.1 Goal Set . . . 6
- 2.2 Goals → Roles → Services (Traceability) . . . 6
- 2.3 Acceptance Criteria . . . 6

3 Roles Model **7**
- 3.1 Roles Table (Updated) . . . 7

4 Agent System Architecture **8**
- 4.1 Component Diagram . . . 8

5 Interaction Model **9**
- 5.1 Protocol Specification (FIPA-style) . . . 9
- 5.2 Message Schema (Headers/Body) . . . 9
- 5.3 Message Sequence Charts . . . 9

6 PAA Algorithms & Behaviour **11**
- 6.1 Planning/Replanning Flowchart . . . 11
- 6.2 PAA Statechart . . . 11
- 6.3 Agent Workflow (End-to-End) . . . 12
- 6.4 Fitness Function & Local Repair . . . 13

7 Services Model **14**

8 Data Specification **15**
- 8.1 Data Dictionary . . . 15
- 8.2 E–R Sketch . . . 15

9 Quality Attributes & Risks **16**
- 9.1 NFR Scenarios . . . 16

10 Acquaintance Model & Summary **17**
10.1 Acquaintance Model 17
10.2 Summary 17

List of Figures

4.1 High-level component/agent architecture. 8

5.1 Plan request protocol. 10
5.2 Disruption handling protocol. 10

6.1 PAA planning and re-planning flow. 11
6.2 PAA behavioural statechart. 12
6.3 Agent workflow from task intake to plan commit. 12

8.1 E–R sketch for MAPS storage. 15

List of Tables

3.1 Table 3.1: Roles Table (updated with Permissions, Knowledge, and I/O & KPIs). 7

1. Methodology & Overview

We adopt the **GAIA** methodology to specify: goals, roles, interactions, agent model, services model, acquaintance model, and internal architectures. The design targets a single-user daily planning MAS with three core agents: *UserProfileAgent (UPA)*, *TaskManagementAgent (TMA)*, and the merged *Planning & Adaptation Agent (PAA)*.

1.1 Design Objectives

- Produce feasible, high-quality schedules under deadlines, dependencies, and availability.
- Re-plan rapidly on disruptions while minimizing schedule churn.
- Learn user-specific energy curves and preference weights to improve fit.

2. Goals & Traceability

2.1 Goal Set

- **G1** Maximize priority satisfaction under constraints.
- **G2** Minimize re-plan latency and plan churn after disruptions.
- **G3** Personalize plans using learned energy/pref models.

2.2 Goals → Roles → Services (Traceability)

Goal	Primary Role(s)	Key Services
G1	PAA, TMA	GeneratePlan, ValidateConstraints, CommitSchedule
G2	PAA	RePlanFast (local repair), GlobalOptimize (GA), DeltaPropose
G3	UPA, PAA	LearnPreferences, ForecastEnergy, PlanWithEnergy

2.3 Acceptance Criteria

- Initial plan for a typical day ($\leq$ 40 tasks) in $\leq$ 2s; local re-plan p50 $\leq$ 150ms; p95 $\leq$ 500ms.
- No hard-constraint violations (deadlines, double booking); conflicts resolved before commit.
- Energy alignment score improves week-over-week (monotone non-decreasing median).

3. Roles Model

3.1 Roles Table (Updated)

Role	Responsibilities (Liveness / Safety)	Permissions	Knowledge	I/O & KPIs
User Profile Agent (UPA)	*Liveness:* $(ReceiveEvents \cdot Update \cdot PublishForecast)^{\omega}$. *Safety:* Forecast $\in [0,1]$; no raw PII exposure.	Read event stream; write model parameters; answer forecast queries.	Energy curve parameters; preference weights; feedback history.	*In:* events, feedback. *Out:* energy_curve, weights. *KPIs:* MAE of energy forecast, feedback acceptance rate.
Task Management Agent (TMA)	*Liveness:* $(Validate \cdot Version \cdot PublishGraph)^{\omega}$. *Safety:* DAG only; immutable IDs.	Full CRUD on tasks; publish versioned graphs; validate dependencies.	Task DAG; critical path; effort/priority metadata.	*In:* task CRUD. *Out:* task_graph (ver). *KPIs:* validation error rate, time-to-graph.
Planning & Adaptation Agent (PAA)	*Liveness:* $InitPlan \cdot (Observe \cdot RePlan)^{*}$. *Safety:* respect hard constraints; no double booking.	Read forecasts/graphs; write schedule store; propose/commit via UI.	Constraint set; schedule model; scoring weights; churn threshold θ.	*In:* plan request, disruption events. *Out:* schedule, delta, rationale. *KPIs:* plan latency (p50/p95), violations=0, churn $\leq \theta$.

Table 3.1: Table 3.1: Roles Table (updated with Permissions, Knowledge, and I/O & KPIs).

4. Agent System Architecture

4.1 Component Diagram

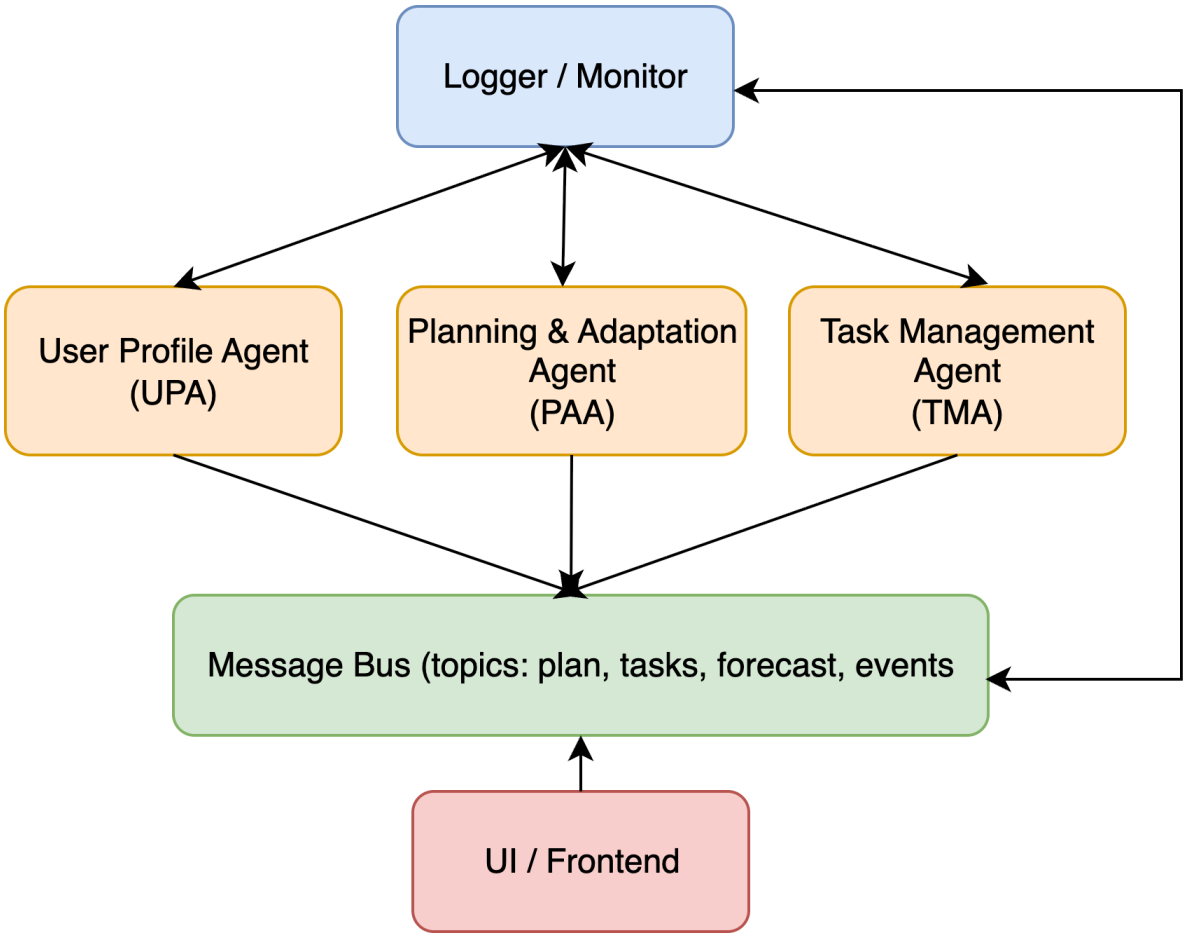


Figure 4.1: High-level component/agent architecture.

5. Interaction Model

5.1 Protocol Specification (FIPA-style)

Protocol	Performatives	Pre/Post Conditions	Timeout/Retry
PlanRequest	REQUEST, QUERY, INFORM, PROPOSE, ACCEPT / REJECT	Pre: Task graph version exists; forecast ready or default. Post: committed schedule or rejection.	2s / 1 retry
Disruption Handling	INFORM, PROPOSE, ACCEPT	Pre: Current schedule exists. Post: updated schedule (delta) committed.	1s / 2 retries
FeedbackLoop	INFORM	Pre: Feedback payload valid. Post: model weights updated; optional forecast refresh.	async

5.2 Message Schema (Headers/Body)

Header Fields	`performative`, `topic`, `conversation_id`, `correlation_id`, `timestamp`
Body (JSON)	`type`, `payload` (e.g., task_graph, energy_curve, schedule, disruption)

5.3 Message Sequence Charts

MS1: Plan Request

MS2: Disruption Handling

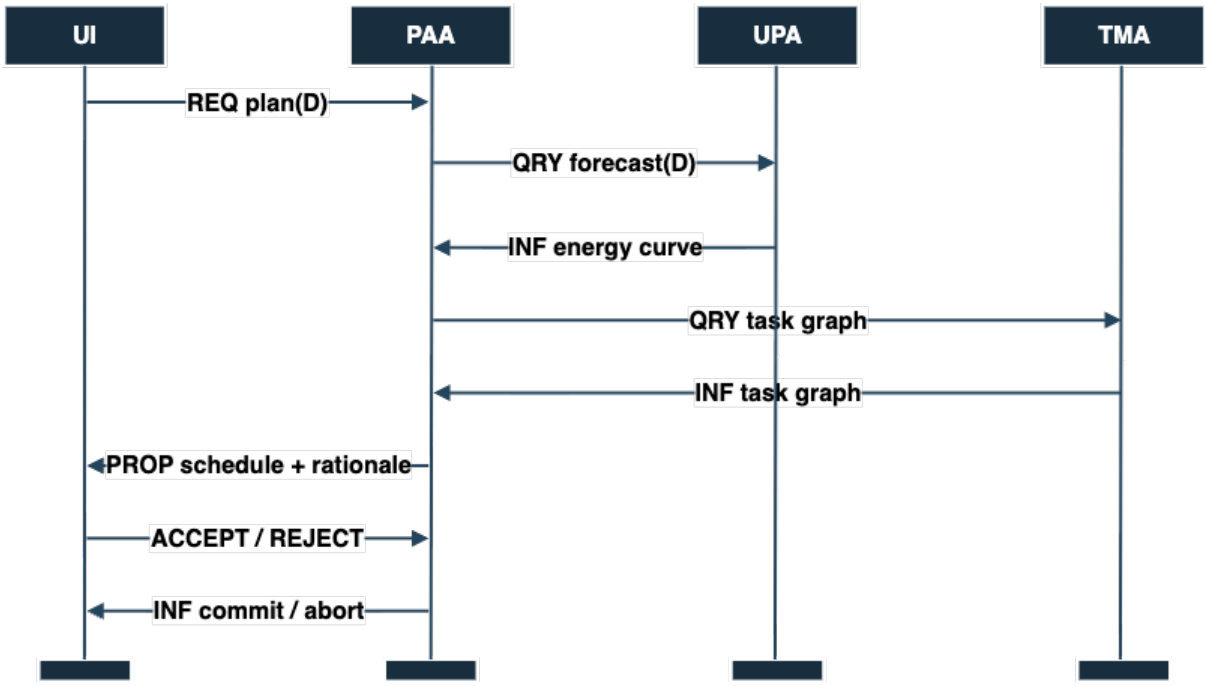


Figure 5.1: Plan request protocol.

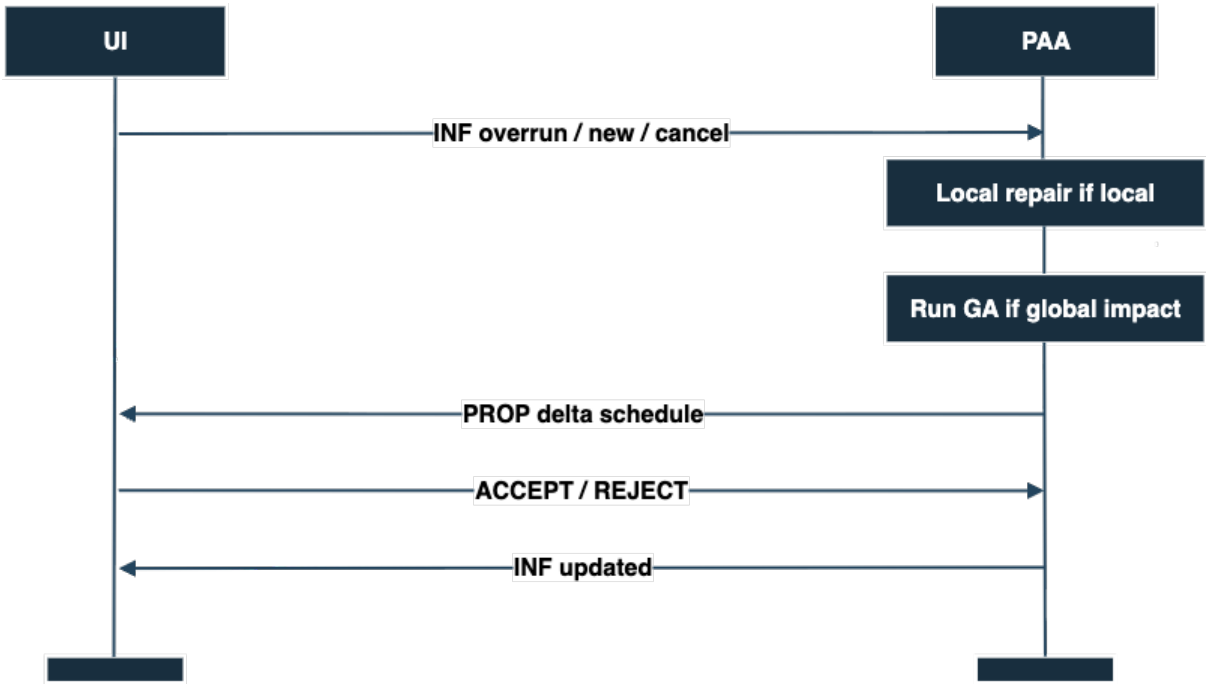


Figure 5.2: Disruption handling protocol.

6. PAA Algorithms & Behaviour

6.1 Planning/Replanning Flowchart

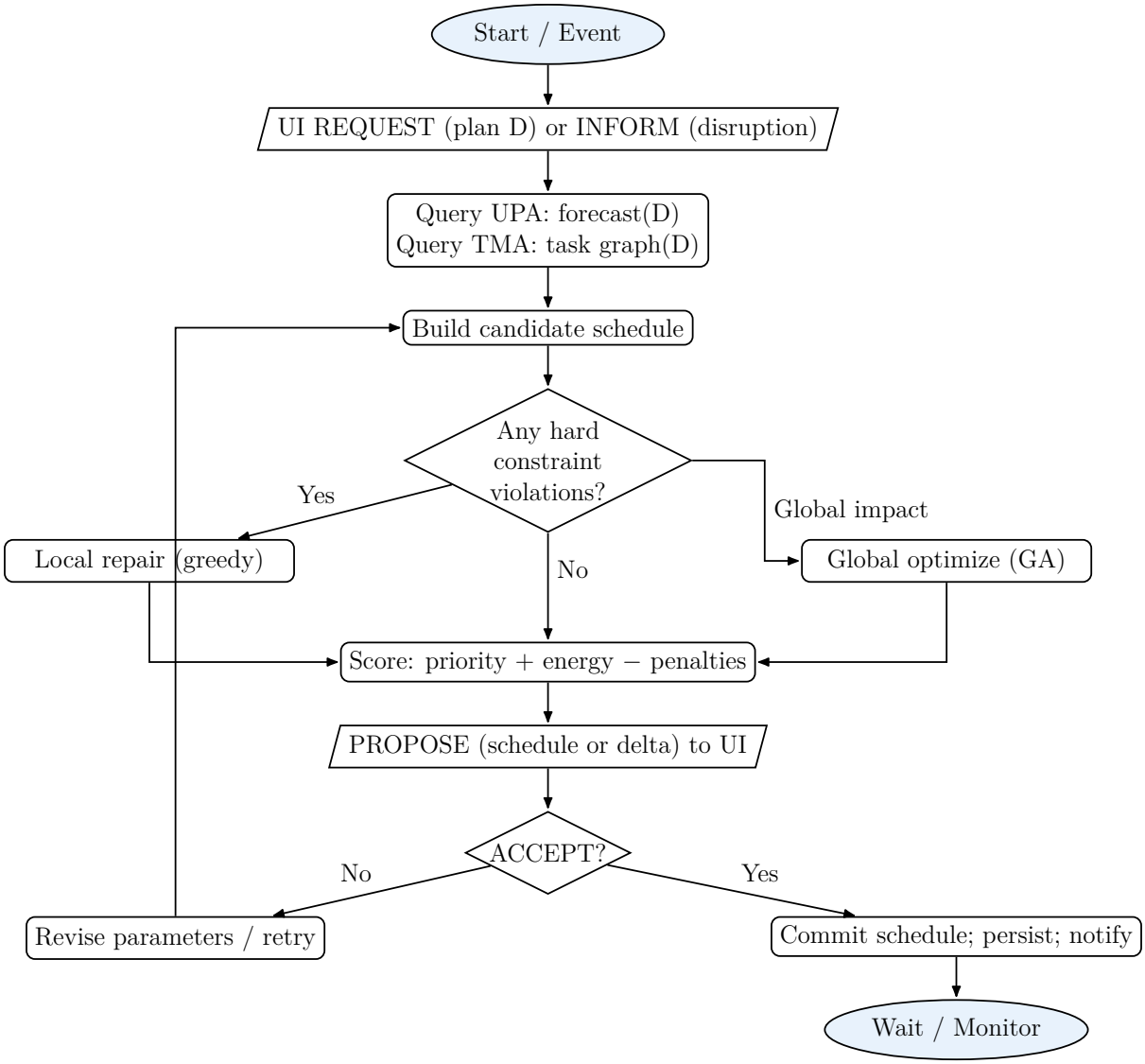


Figure 6.1: PAA planning and re-planning flow.

6.2 PAA Statechart



Figure 6.2: PAA behavioural statechart.

6.3 Agent Workflow (End-to-End)

Figure 6.3: Agent workflow from task intake to plan commit.

6.4 Fitness Function & Local Repair

Fitness Components

Component	Weight Range	Notes
Priority Satisfaction	$w_p \in [0.3, 0.6]$	Task importance, deadlines, criticality path
Energy Alignment	$w_e \in [0.2, 0.5]$	Match energy curve to task difficulty
Travel/Transition Penalty	$w_t \in [0.0, 0.2]$	Minimize context switches/travel time
Constraint Penalty	$w_c \in [0.3, 0.7]$	Hard constraints get large negative penalties

Local Repair (pseudo-code)

Listing 6.1: Greedy local repair at disruption

```
def local_repair(schedule, disruption):
    impacted = find_impacted_blocks(schedule, disruption)
    for block in sort_by_urgency(impacted):
        slots = enumerate_feasible_slots(schedule, block.task)
        best = argmax(slots, key=lambda s: score_delta(schedule, block, s))
        if best and respects_hard_constraints(best):
            schedule = place(schedule, block, best)
        else:
            return escalate_to_global(schedule, impacted)
    return schedule
```

7. Services Model

Service	Inputs	Pre-Conditions	Post-Conditions / Failure Modes
GeneratePlan	task_graph, energy_forecast, availability	Valid DAG; availability defined	Schedule persisted; conflicts=0. Fail: invalid graph, timeout.
RePlanFast	disruption_event	Current schedule exists	Delta applied; churn $\leq \theta$. Fail: escalate to GA.
GlobalOptimize	task_graph, energy_forecast	Impact is global	New schedule proposed. Fail: timeout $\Rightarrow$ fallback template.
LearnPreferences	events, feedback	Data window available	Updated weights; monotonic constraints enforced.

8. Data Specification

8.1 Data Dictionary

Entity	Fields	Key
Task	id, title, deadline, effort, priority, status, depends_on (nullable)	pk:id
ScheduleBlock	id, task_id, start, end, confidence	pk:id, fk:task_id
Preference	id, key, value (float), updated_at	pk:id
Event	id, ts, type, payload(json)	pk:id

8.2 E–R Sketch

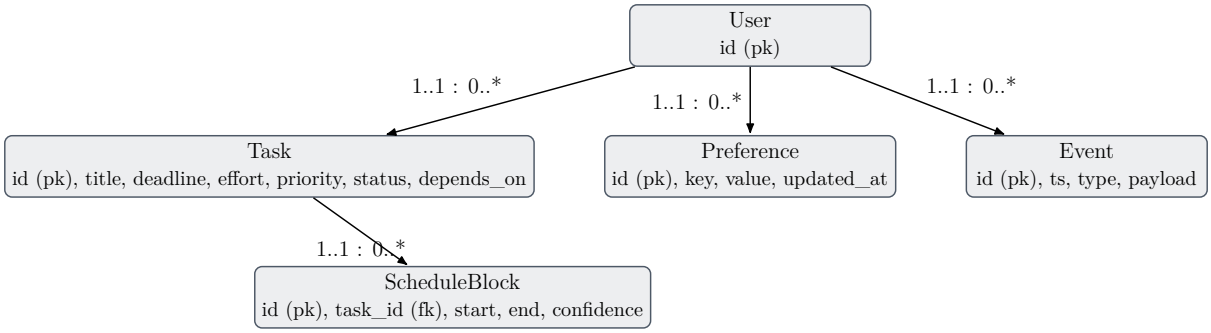


Figure 8.1: E–R sketch for MAPS storage.

9. Quality Attributes & Risks

9.1 NFR Scenarios

Attribute	Scenario	Target
Performance	Initial plan for ≤ 40 tasks	≤ 2s
Performance	Local re-plan after single overrun	p50 $\leq$ 150ms; p95 $\leq$ 500ms
Reliability	Crash during planning; on restart	Recover last committed schedule
Privacy	All data local; encrypted at rest	AES-256; minimal payloads over bus

10. Acquaintance Model & Summary

10.1 Acquaintance Model

UI ↔ PAA (plan lifecycle), UI ↔ TMA (task CRUD), UI ↔ UPA (feedback), PAA ↔ UPA (forecasts), PAA ↔ TMA (graphs), All → Logger.

10.2 Summary

This extended GAIA design provides traceability from goals to roles and services, formalizes protocols and message schemas, details PAA behaviour via flowchart and statechart, and adds an end-to-end agent workflow for implementation (SPADE/JADE) and assessment.