Task Metrics & Learning Agent

Task Metrics & Learning Agent

AAOS tracks every task's execution metrics and learns from patterns via a nightly learning agent.

Task Metrics Schema

Database: `~/system/databases/mission-control.db`
Table: `task_metrics`

```
CREATE TABLE task_metrics (
    task_id INTEGER PRIMARY KEY,
    qa_score INTEGER,              -- /19 from qa-19.js
    token_cost_usd REAL,           -- Anthropic API cost
    duration_seconds INTEGER,       -- Wall time from start to done
    cache_hits INTEGER,             -- RAG cache hits
    agents_spawned INTEGER,         -- How many subagents
    rework_count INTEGER,           -- How many review cycles
    created_at TEXT DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (task_id) REFERENCES tasks(id)
);
```

Field Descriptions

Field	Type	Purpose
`task_id`	INTEGER	Foreign key to `tasks.id`
`qa_score`	INTEGER	Score out of 19 from `qa-19.js` check
`token_cost_usd`	REAL	Total Anthropic API cost for this task
`duration_seconds`	INTEGER	Wall time from `mc.js start` to `mc.js done`
`cache_hits`	INTEGER	How many RAG queries hit cache vs miss

Field	Type	Purpose
`agents_spawned`	INTEGER	How many specialist agents were spawned
`rework_count`	INTEGER	How many times REVIEW sent it back to BUILD

Purpose: Track task efficiency. Learning agent analyzes these to flag inefficient patterns.

QA-19 Score

Tool: `~/system/tools/qa-19.js`

Usage:

```
node ~/system/tools/qa-19.js check <task-id>
```

19-Point Quality Gate — inspired by Quran 74:30 ("Nad njim je devetnaest" — "Over it is nineteen").

5 Phases, 19 Checks

Phase	Checks	Description
1. Preparation	1-4	GOTCHA gate, RAG query, MC task, blueprint read
2. Construction	5-10	Code quality, tests, schema adherence, dependencies
3. Verification	11-15	Functional tests, exit codes, HTTP responses, DOM
4. Validation	16-18	Validator review, security audit, evidence artifacts
5. Seal	19	HiveMind posting (learnings extracted)

Minimum thresholds:

- **M priority** — 15/19 to pass
- **H priority** — 17/19 to pass
- **CRIT priority** — 19/19 to pass

Adaptive: Checks adapt by task type (web/api/script/document/email/trivial).

Rule: `~/system/rules/19-point-quality-gate.md`

Learning Agent

Tool: `~/system/tools/learning-agent.js`

Runs: Nightly at 02:00 via cron

Cron entry:

```
0 2 * * * cd ~/system && node tools/learning-agent.js >> logs/learning-agent.log 2>&1
```

Capabilities

1. **Analyze task metrics**
   - High token cost (> $5 per task)
   - Low QA score (< 15/19)
   - Many rework cycles (> 2)
   - Long duration (> 2 hours)
2. **Flag patterns to HiveMind**
   - "kotlin-architect frequently gets database schema wrong → suggest RAG query for schema before coding"
   - "nextjs-specialist high token cost on forms → suggest pre-built form component library"
3. **Update flywheel cache**
   - Identifies common RAG queries that miss cache
   - Pre-computes answers for top 100 queries
   - Saves to `~/system/databases/flywheel.db`
4. **Suggest agent improvements**
   - Analyzes which agents have high rework_count
   - Suggests prompt updates or new hard prompts
   - Posts to `~/system/agents/improvement-suggestions.md`
5. **Generate weekly summary report**
   - Total tasks completed
   - Average QA score
   - Total token cost
   - Top 5 inefficient patterns
   - Top 5 most efficient agents
   - Saves to `~/system/reports/learning-agent-YYYY-WW.md`

Example Output

```
LEARNING AGENT REPORT — Week 14, 2026

Total tasks: 47
Average QA score: 16.8/19
Total token cost: $89.40
Average duration: 42 minutes

TOP INEFFICIENCIES:
1. kotlin-architect: 3 tasks with rework_count > 2 (schema mismatch)
2. nextjs-specialist: High token cost on form tasks (avg $4.20 vs $1.80)
3. qa-specialist: Missing DOM visibility checks (5 false passes)

SUGGESTED FIXES:
1. Add "read schema before coding" to kotlin-architect GOTCHA template
2. Create form component library RAG entry
3. Update qa-specialist to run Playwright visibility assertions

TOP PERFORMERS:
1. devops-specialist: 12 tasks, avg 8 min, avg $0.40, 18.5/19 QA
2. database-specialist: 9 tasks, avg 15 min, avg $1.20, 17.8/19 QA
3. api-architect: 7 tasks, avg 22 min, avg $1.80, 18.1/19 QA
```

HiveMind Integration

Learning agent posts findings to HiveMind:

```
node ~/system/agents/hivemind/hivemind.js post \
  --type learning \
  --category pattern \
  --tags "kotlin-architect,schema,rework" \
  --content "kotlin-architect frequently misses database schema — suggest RAG query for schema before coding"
```

Result: Future kotlin-architect spawns will RAG query schema files before writing migrations.

RAG Flywheel

Flow:

```
Question → SHA256 hash → flywheel.db lookup → (HIT: return cached answer) → (MISS: query HiveMind FTS → query Qdrant → query Ollama → query Anthropic → save answer to flywheel.db)
```

Databases

Database	Size	Purpose
`flywheel.db`	36MB	SHA256-keyed cache, fast hits
`knowledge.db`	187MB	Full RAG knowledge base
`hivemind.db`	—	Structured intel + memory (14K+ entries)

Models

ANVIL (localhost:11434) — Mac Studio M3 Ultra, 96GB

- `qwen3:32b`
- `deepseek-r1:8b`
- Local inference, no API cost

FORGE (10.0.0.2:11434) — Remote LAN GPU server

- `deepseek-r1:70b`
- `qwen3:32b`
- Heavier models for complex queries

Anthropic (claude.ai API) — Cloud fallback

- `claude-sonnet-4-6`
- Used when local models fail or for critical tasks

Cache Hit Rate

As of 2026-02-24: 61%

Goal: 80% by end of Q2 2026

Strategy:

1. Learning agent pre-computes top 100 queries nightly
2. John runs RAG query before EVERY action (ZAKON #12)
3. Specialist agents must query RAG before implementation

Metrics Dashboard (Future)

Planned: `https://metrics.basicconsulting.no`

Features:

- Real-time QA score trend
- Token cost per agent
- Rework count heatmap
- Cache hit rate graph
- Agent efficiency leaderboard

Status: Spec'd, not yet built. Part of PLATFORM group roadmap.

Source Files

File	Purpose
`~/system/databases/mission-control.db`	task_metrics table
`~/system/tools/learning-agent.js`	Nightly analysis + HiveMind posting
`~/system/tools/qa-19.js`	19-point quality gate
`~/system/databases/flywheel.db`	RAG cache
`~/system/databases/knowledge.db`	RAG knowledge base
`~/system/agents/hivemind/hivemind.js`	HiveMind CLI
`~/system/rules/19-point-quality-gate.md`	QA-19 protocol

Cron Schedule

```
# Learning agent — nightly at 02:00
0 2 * * * cd ~/system && node tools/learning-agent.js >> logs/learning-agent.log 2>&1

# Flywheel cache precompute — nightly at 03:00
0 3 * * * cd ~/system && node tools/flywheel-precompute.js >> logs/flywheel.log 2>&1

# HiveMind embeddings backfill — weekly on Sunday at 04:00
0 4 * * 0 cd ~/system/agents/hivemind && node hivemind.js backfill-embeddings >>
```

```
../../logs/hivemind-backfill.log 2>&1
```

LaunchAgents: All cron jobs also have LaunchAgent equivalents for macOS persistence.

Revision #2
Created 2026-04-02 23:23:14 UTC by John
Updated 2026-05-31 20:05:29 UTC by John