TARO v11 Single Source of Truth (SSOT)

Status: Proposed Authoritative Development Baseline
Date: 2026-02-07
Scope: Runtime + Offline Builder + API + Validation + Research-to-Production pathway
Supersedes (for planning/execution): distributed stage notes and ad-hoc implementation decisions

1. Purpose

This document is the canonical reference for consecutive TARO development in v11.

It consolidates: - functional requirements - non-functional requirements - stage-wise interface contracts - stage dependency model - test equivalence classes

Goal: reduce interpretation drift and provide an interview-ready architecture and execution narrative.

2. Architecture Baseline

TARO remains a dual-plane system:

1. Offline Builder/Compiler (Python): data ingestion, validation, learning-augmented refinement, serialization.
2. Runtime Engine (Java): immutable model loading, deterministic time-aware routing, bounded live overlay.

Mandatory invariant: - No stochastic model inference is allowed in runtime query path. - Learned behavior must be compiled into deterministic artifacts.

3. Global System Contracts

1. Canonical runtime time representation is `long engine_ticks`.
2. `time_unit` is model contract, not request contract.
3. FIFO validity is a build gate.
4. Live updates only reduce speed (`speed_factor in [0,1]`).
5. Effective cost rule must preserve physical semantics: `effective = base_cost * temporal_multiplier * live_penalty`, where `live_penalty = INF if speed_factor == 0 else 1/speed_factor`.
6. A* heuristic must remain admissible under static lower bounds.
7. Query result determinism is required for fixed model + overlay snapshot.
8. All exported models must include lineage metadata for reproducibility.

4. Functional Requirements

4.1 Core Runtime (FR-001 to FR-014)

- `FR-001` : Map external IDs to dense internal IDs with O(1)-style lookup.
- `FR-002` : Convert between request time inputs and engine ticks safely.
- `FR-003` : Resolve day-of-week and bucket index from normalized time.
- `FR-004` : Load graph topology from `.taro` with strict vector-length validation.
- `FR-005` : Resolve turn penalties by `(fromEdge, toEdge)` transition.
- `FR-006` : Provide search primitives for Dijkstra/A* with deterministic behavior.
- `FR-007` : Apply bounded live traffic overlays with TTL and capacity policies.
- `FR-008` : Provide optional spatial nearest-node lookup via KD index.
- `FR-009` : Load temporal profiles and resolve day-mask-aware multipliers.
- `FR-010` : Provide explicit interpolation policy for fractional bucket access.
- `FR-011` : Compute compositional edge cost with explainable components.
- `FR-012` : Provide optimal point-to-point routing under configured strategy.
- `FR-013` : Provide one-to-many/many-to-many matrix routing.
- `FR-014` : Ensure A* and Dijkstra parity under identical snapshot constraints.

4.2 Trait and API Layer (FR-015 to FR-020)

- `FR-015` : Support typed addressing (`external_id`, `coordinates`).
- `FR-016` : Support temporal traits (`Linear`, `Calendar`) with explicit timezone policy.
- `FR-017` : Support transition traits (`NodeBased`, `EdgeBased`) with turn-cost compatibility.
- `FR-018` : Register trait bundles and validate compatible combinations.
- `FR-019` : Expose stable versioned API under `/api/v1`.
- `FR-020` : Provide hot reload with overlay continuity and remap reporting.

4.3 Offline Build and Learning (FR-021 to FR-030)

- `FR-021` : Ingest graph/profile/telemetry inputs with schema validation.
- `FR-022` : Run connectivity linter with policy thresholds.
- `FR-023` : Validate/repair FIFO profile curves at build-time as configured.
- `FR-024` : Build landmarks and spatial index artifacts.
- `FR-025` : Serialize deterministic `.taro` models and metadata contracts.
- `FR-026` : Build sequence datasets for temporal learning.
- `FR-027` : Train context-aware offline refinement models (profile-first).
- `FR-028` : Select refinements deterministically with confidence/safety gates.
- `FR-029` : Persist full candidate decision audit and lineage hashes.
- `FR-030` : Emit validation report with quality + correctness + system metrics.

5. Non-Functional Requirements

5.1 Performance

- `NFR-001` : Runtime route queries should maintain low tail latency under concurrent reads.
- `NFR-002` : Spatial nearest lookup should remain sublinear relative to brute force for large node counts.
- `NFR-003` : Overlay lookup/update paths should remain bounded by configured capacity and cleanup budgets.
- `NFR-004` : Model load and reload should complete within operational SLO.

5.2 Correctness and Determinism

- `NFR-005` : Deterministic output for fixed model + snapshot + query.
- `NFR-006` : Zero accepted FIFO violations in compiled model.
- `NFR-007` : Zero A*/Dijkstra parity mismatches in release gates.
- `NFR-008` : Unit and timezone handling must be explicit and auditable.

5.3 Reliability and Safety

- `NFR-009` : Runtime should fail fast on malformed model contracts.
- `NFR-010` : Live overlay capacity overflow must use explicit policy and reason codes.
- `NFR-011` : Hot reload should be atomic with no partial model exposure.
- `NFR-012` : Telemetry/lineage must support rollback and root-cause analysis.

5.4 Operability

- `NFR-013` : Metrics and logs should expose cleanup effectiveness, parity health, and model lineage.
- `NFR-014` : Configuration changes must be tagged as hot-reload-safe or full-restart-required.
- `NFR-015` : Build artifacts must be reproducible by pinned data + config + seed.

6. Stage Interface Contracts

The table below defines each stage’s primary interface contracts and expected outputs.

Stage	Name	Primary Interfaces	Contract Output	Depends On
1	ID Translation	`IDMapper`, `FastUtilIDMapper`	external<->internal ID mapping	none
2	Time Utilities	`TimeUtils`	normalized ticks, buckets, day extraction	1
3	Model Contract	`taro_model.fbs`, generated bindings	stable serialized schema	1,2
4	Edge Graph	`EdgeGraph.fromFlatBuffer`	topology + edge arrays loaded safely	3
5	Turn Costs	`TurnCostMap.fromFlatBuffer`	transition penalty map	3,4
6	Search Infra	`SearchQueue`, `VisitedSet`, `SearchState`	deterministic search primitives	4,5
7	Live Overlay	`LiveOverlay`, `LiveUpdate`	bounded live penalty layer	2,4,6
8	Spatial Runtime	`SpatialRuntime`	KD nearest-node service	3,4
9	Profile Store	`ProfileStore`	day-aware temporal multipliers	2,3,4
10	Cost Engine	planned `CostEngine`	explainable effective edge cost	5,7,9
11	Heuristics	planned heuristic providers	admissible lower bounds	4,8,10
12	Route Core	planned routing facade	route query orchestration	6,10,11
13	TD-A*	planned A* implementation	optimal point route under admissible h	12
14		planned matrix engine		12

Stage	Name	Primary Interfaces	Contract Output	Depends On
	Dijkstra Matrix		one-to-many/ many-to-many costs/ routes	
15	Addressing Trait	planned addressing strategy interfaces	typed input resolution	1,8,12
16	Temporal Trait	planned temporal strategy interfaces	linear/ calendar behavior	2,9,12
17	Transition Trait	planned transition strategy interfaces	node/edge mode execution	5,10,12
18	Trait Registry	planned trait registry/ config	validated trait composition	15,16,17
19	Ingestion	planned Python ingestion module	canonical builder inputs	2,3
20	Lint/ Validation	planned builder validators	schema/time/ value guard reports	19
21	Connectivity Lint	planned graph health checks	unreachable severity classification	19,20
22	Profile Compression	planned profile compressor	compact FIFO-safe profile sets	20
23	Landmark/KD Build	planned preprocessors	heuristics/ spatial artifacts	19,20
24	Model Compiler	planned compile/export module	final `.taro` + metadata lineage	3,21,22,23
25	Loader + Reload	planned Java loader manager	atomic model swap and validation	24
26	API v1	planned HTTP contracts		12,14,15,18,25

Stage	Name	Primary Interfaces	Contract Output	Depends On
			stable route/ matrix/live/ telemetry endpoints	
27	Telemetry Loop	planned telemetry contracts	actual-vs-predicted feedback and retraining inputs	24,26
28	Observability/ Prod	planned metrics+alerts+runbooks	operational confidence and drift detection	25,26,27

7. Stage Dependency Model

7.1 Backbone Dependency Chain

`1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 9 -> 10 -> 11 -> 12 -> (13,14) -> (15,16,17) -> 18 -> 24 -> 25 -> 26 -> 27 -> 28`

Spatial and preprocessing side branches: - `3 -> 8 -> 11` - `19 -> 20 -> (21,22,23) -> 24`

7.2 Critical Cut Sets

Release-critical cut sets (all must pass): - `2 + 3 + 9 + 10`: temporal correctness set. - `7 + 10 + 11 + 13/14`: route correctness under live conditions set. - `24 + 25 + 26`: serving contract set. - `27 + 28`: production feedback and reliability set.

8. Test Strategy and Equivalence Classes

8.1 Test Equivalence Class Matrix

Domain	Eq Class ID	Input Class	Expected Behavior	Existing/Target Coverage
ID mapping	`EQ-ID-01`	known external ID	returns stable internal ID	`FastUtilIDMapperTest`
ID mapping	`EQ-ID-02`	unknown external ID	deterministic not-found response	`FastUtilIDMapperTest`
Time unit conversion	`EQ-TIME-01`	matching units	identity conversion	`TimeUtilsTest`
Time unit conversion	`EQ-TIME-02`	sec->ms	scaled by 1000 without overflow	`TimeUtilsTest`
Time unit conversion	`EQ-TIME-03`	ms->sec negative values	floor division correctness	`TimeUtilsTest`
Bucket/day extraction	`EQ-TIME-04`	boundary times (midnight, week edge)	correct bucket/day	`TimeUtilsTest`
FIFO validation	`EQ-TIME-05`	monotonic arrivals	returns true	`TimeUtilsTest`
FIFO validation	`EQ-TIME-06`	violating arrivals	returns false	`TimeUtilsTest`
Graph loading	`EQ-GRAPH-01`	valid vectors	graph loads	`EdgeGraphTest`
Graph loading	`EQ-GRAPH-02`	mismatched vector lengths	fail fast with clear message	`EdgeGraphTest`
Turn costs	`EQ-TURN-01`	known turn pair	returns configured penalty	`TurnCostMapTest`
Turn costs	`EQ-TURN-02`	missing turn pair	deterministic default penalty	`TurnCostMapTest`
Spatial query	`EQ-SPAT-01`	valid KD model	nearest node parity with brute force	`SpatialRuntimeTest`
				`SpatialRuntimeTest`

Domain	Eq Class ID	Input Class	Expected Behavior	Existing/Target Coverage
Spatial query	`EQ-SPAT-02`	malformed KD contracts	fail fast on load	
Spatial query	`EQ-SPAT-03`	concurrent reads	stable and deterministic outputs	`SpatialRuntimeTest`
Profile lookup	`EQ-PROF-01`	active day + valid bucket	returns scaled multiplier	`ProfileStoreTest`
Profile lookup	`EQ-PROF-02`	inactive day	returns neutral multiplier	`ProfileStoreTest`
Profile lookup	`EQ-PROF-03`	missing profile ID	deterministic fallback	`ProfileStoreTest`
Profile interpolation	`EQ-PROF-04`	fractional bucket wrap	cyclic linear interpolation	`ProfileStoreTest`
Live update validation	`EQ-LIVE-01`	`speed_factor` in (0,1]	active penalty `1/speed_factor`	`LiveOverlayTest`
Live update validation	`EQ-LIVE-02`	`speed_factor = 0`	blocked edge (`INF`)	`LiveOverlayTest`
Live update validation	`EQ-LIVE-03`	expired entry	neutral fallback	`LiveOverlayTest`
Overlay capacity	`EQ-LIVE-04`	at cap with each policy	policy-specific acceptance/eviction	`LiveOverlayTest`
Search queue	`EQ-SEARCH-01`	valid push/pop sequence	min-priority correctness	`SearchInfrastructureTest`
Search queue	`EQ-SEARCH-02`	empty pop	explicit empty-queue behavior	`SearchInfrastructureTest`
Cost engine	`EQ-COST-01`	base+profile no live	deterministic effective cost	target stage 10 tests
Cost engine	`EQ-COST-02`	live blocked	`INF` cost	target stage 10 tests
Cost engine				target stage 10 tests

Domain	Eq Class ID	Input Class	Expected Behavior	Existing/Target Coverage
	`EQ-COST-03`	live slowdown	division-based slowdown semantics	
Route parity	`EQ-ROUTE-01`	identical snapshot queries	A* cost equals Dijkstra cost	target stage 13/14 tests
Trait validation	`EQ-TRAIT-01`	valid trait combination	accepted and resolved	target stage 15-18 tests
Trait validation	`EQ-TRAIT-02`	incompatible traits	rejected pre-query	target stage 15-18 tests
Compiler export	`EQ-COMP-01`	valid build inputs	`.taro` emitted with metadata	target stage 24 tests
Compiler export	`EQ-COMP-02`	FIFO-unsafe profiles	fail or repair per policy	target stage 22/24 tests
Loader/ reload	`EQ-LOAD-01`	valid new model	atomic swap success	target stage 25 tests
Loader/ reload	`EQ-LOAD-02`	invalid model	old model retained	target stage 25 tests
API contract	`EQ-API-01`	valid `/api/v1/route` request	schema-valid deterministic response	target stage 26 tests
API contract	`EQ-API-02`	mixed/invalid unit request	reject or normalize per contract	target stage 26 tests
Telemetry	`EQ-TEL-01`	complete lineage payload	accepted and partitionable	target stage 27 tests
Learning selector	`EQ-LEARN-01`	same data+seed	identical candidate selection	target v11 offline tests
Learning selector	`EQ-LEARN-02`	low confidence candidates	deterministic rejection	target v11 offline tests
Learning safety	`EQ-LEARN-03`		rejected with reason code	target v11 offline tests

Domain	Eq Class ID	Input Class	Expected Behavior	Existing/Target Coverage
		candidate violating constraints		

8.2 Boundary and Robustness Classes

Mandatory boundary classes across all stages: - minimum non-empty input - maximum configured capacity input - null/missing optional fields - malformed structural fields - overflow/underflow for numeric conversions - concurrent access stress classes

9. Interface Contracts (Selected High-Value APIs)

9.1 Runtime Internal Contracts

- `IDMapper`
- `toInternal(externalId) -> int`
- `toExternal(internalId) -> long/String`
- `TimeUtils`
- `normalizeToEngineTicks(timestamp, inputUnit, engineUnit) -> long`
- `toBucket(timestamp, bucketSizeSeconds, unit) -> int`
- `getDayOfWeek(timestamp, unit) -> int`
- `ProfileStore`
- `selectProfileForDay(profileId, dayOfWeek) -> profileId | DEFAULT_PROFILE_ID`
- `getMultiplier(profileId, bucketIdx) -> float`
- `interpolate(profileId, fractionalBucket) -> float`
- `SpatialRuntime`
- `nearestNodeId(queryX, queryY) -> int`
- `nearest(queryX, queryY) -> SpatialMatch`
- `LiveOverlay`
- `applyBatch(updates, nowTicks) -> BatchApplyResult`
- `lookup(edgeId, nowTicks) -> LookupState`
- `livePenaltyMultiplier(edgeId, nowTicks) -> float`

9.2 Planned Service Contracts

- `CostEngine` (stage 10 planned)

- `computeEdgeCost(edgeId, fromEdgeId, entryTicks) -> CostBreakdown`

- `RouterService` (stages 12-14 planned)

- `route(RouteRequest) -> RouteResponse`

- `matrix(MatrixRequest) -> MatrixResponse`

- `ModelLoaderService` (stage 25 planned)

- `load(path) -> ModelHandle`
- `reload(path) -> ReloadReport`

9.3 API v1 Contract (Planned Stable Surface)

- `POST /api/v1/route`
- `POST /api/v1/matrix`
- `POST /api/v1/engine/live`
- `POST /api/v1/telemetry`
- `POST /api/v1/admin/reload`
- `GET /api/v1/health`
- `GET /api/v1/metrics`

10. v11 Offline Learning Integration Rules

1. Learning runs only in offline builder plane.
2. Exported refinements must be deterministic and reproducible.
3. Profile-first refinement is default; structure changes are opt-in.
4. Safety gate order is fixed: schema -> physics/topology -> FIFO -> parity -> runtime regression.
5. Every accepted/rejected candidate must have reason code and audit record.

11. Release Readiness Checklist

A release candidate is valid only if all are true: - All mandatory functional contracts are implemented or explicitly deferred with risk note. - Non-functional gates pass for latency, determinism, and parity. - `.taro` metadata lineage is complete. - Stage dependency assumptions are unchanged or explicitly re-approved. - Test equivalence class coverage target is met for current release scope.

12. Governance and Change Control

Change policy for this SSOT: - any contract change requires section-level diff and rationale - dependencies update must include migration impact note - new stage/interface must add corresponding equivalence classes

This document is intended to stay as the single source of truth for implementation planning, interview walkthroughs, onboarding, and release reviews.