MOSAIC — Middle-Out Structured, Auditable, Iterative Composition

Version: 0.1
Owner: Phaneron / Delos substrate

---

0) One-page vision

MOSAIC is the "imagination engine" for Phaneron: a representation-first system that composes and decomposes ideas as graphs, not tokens. It builds outputs **from the middle out**—starting at the desired result, growing a minimal skeleton, and recursively expanding only where **Value-of-Information (VoI)** is highest. The graph is the source of truth; text, tables, charts, images, audio, and video are **renderings** with receipts and deterministic replays.

Why it matters

- **Explainable by construction:** every sentence, pixel, or frame traces to nodes/edges and receipts.
- **Efficient & fast:** VoI scheduling avoids bloated context; tacit/explicit switching minimizes compute.
- **Multimodal, not model-bound:** any-in → any-out via adapters; language is just one lens.
- **Deterministic & auditable:** same inputs+seeds ⇒ same output; champion-challenger learning under receipts.

Top user moments

- Watch ideas flower in waves (skeleton → sub-skeletons) and lock into final form with a single click.
- Edit one claim; see the graph propagate changes everywhere; re-render deterministically.
- Replay the "Rechenweg" for any section, or relight/restage a captured scene from its perception graph.

---

1) Goals & Non-goals

Goals

1. Representation-first creation across modalities (text, tables, charts, images, audio, video).
2. Middle-out planning with measurable coverage and confidence.
3. VoI-aware context assembly (budgeted tokens/latency/energy).
4. Deterministic rendering + receipts for provenance and replays.
5. Live, comprehensible UX without flooding users with internals.
6. Instrumentation for self-improvement (champion-challenger, ablations, A/B receipts).

Non-goals (v0.x)

- Bit-exact media reconstruction without residuals.
- Fully autonomous agency or long-horizon tasking.
- Replacing all generative models; we minimize and bound them.

---

2) Key principles

- **Graph over tokens:** ideas live as nodes/edges; tokens are late-stage renderings.
- **Middle-out recursion:** every subgoal is a new "center" that expands only as needed.
- **Tacit ↔ Explicit:** heavy traces are latent by default; expand on demand (teach/audit/fragility).
- **Receipts-or-it-didn't-happen:** every nontrivial step produces a signed, replayable receipt.
- **Amber before wrong:** uncertainty is surfaced; contradictions block rendering until resolved or justified.
- **Any-in → any-out:** modality adapters do parse↔render; round-trip (R^2T^2) is measured.

---

3) Glossary

- **Result node (R):** the desired outcome + acceptance tests.
- **Obligation:** must-cover requirement tied to R (e.g., section, claim, constraint, test).
- **Subgoal:** decomposable target spawned from an obligation.
- **Invariant:** a property that must hold (units, bounds, referential integrity).
- **Evidence binding:** citation/provenance supporting a claim.
- **Receipt:** signed record (inputs, transforms, seeds, outputs, approvals).
- **VoI score:** expected utility of including/expanding a slice under a budget.
- **Amber:** state of flagged uncertainty/contradiction awaiting resolution.

- **R^2T^2:** render-to-recover round-trip score for adapters.

---

4) System architecture

Core components

- **Planner (MOSAIC Core):** middle-out loop → skeletons, subgoals, expansions.
- **Scheduler (Apollo):** VoI-aware node selection under budget.
- **Critic (Artemis/CDA):** coverage, contradiction, and risk checks; champion-challenger tests.
- **Substrate (Delos):** graph store (nodes/edges), receipts store, policies, provenance.
- **Adapters:** parsers (input→graph) and renderers (graph→artifact) per modality.
- **Policy Engine:** safety, privacy membranes, licensing, determinism.
- **Telemetry/Lab:** ablations, A/B, energy/token/latency metering.

High-level flow

1) Create R with acceptance tests.
2) Skeleton pass → obligations.
3) VoI scheduler picks next node; Planner expands (subgoals, invariants, questions, evidence).
4) Critic evaluates coverage/conflicts; may amber, merge (aha-merge), or request evidence.
5) Repeat until acceptance tests pass; render deterministically; store receipts.
6) Optional: counterfactuals/re-renders; learn from telemetry.

---

5) Data model (Delos)

Node types (examples)

- `Result`, `Section`, `Claim`, `Question`, `Evidence`, `Invariant`, `Subgoal`, `Asset(Image|Audio|Video|Table|Chart)`, `Layout`, `Style`, `Camera`, `Lighting`, `Material`, `Pose`.

Edge types

- `contains`, `supports`, `depends_on`, `contradicts`, `verifies`, `aligns_with(time|space)`, `derived_from`, `same_as`.

Receipt (JSON sketch)

```
{
  "id": "rcpt_01F...",
  "time": "2025-10-30T12:34:56Z",
  "actor": "mosaic-core@v0.1",
  "inputs": {"nodes": ["nR","nC1"], "seeds": "0x9e…"},
  "transform": {"name": "expand_subgoals", "params": {"policy": "operate"}},
  "outputs": {"nodes_created": ["nS1","nS2"], "edges": ["e1","e2"]},
  "metrics": {"tokens": 412, "latency_ms": 240, "energy_j": 1.2},
  "approvals": ["policy_ok"],
  "hash": "sha256:…"
}
```

---

6) Core algorithms

6.1 Middle-out loop (pseudocode)

```
function MOSAIC(goal R, budget B):
  init_graph(R); add_acceptance_tests(R)
  obligations <- infer_minimal_obligations(R)
  while not acceptance_satisfied(R) and B.remaining > 0:
    n <- select_next_node(obligations, graph, B)
    expand(n)  // propose subgoals, invariants, questions, evidence bindings
    critique(n) // coverage, contradictions, policy
    if can_merge(): aha_merge()
    update_coverage()
    B.consume(cost(n))
  assert acceptance_satisfied(R)
  return deterministic_render(graph, seeds)
```

6.2 VoI scheduler (selection)

`U(slice) = P(relevance) * (Δaccuracy + Δlatency_save) − λ_tok*tokens − λ_stale*staleness − risk_penalties`

- Representation choice is part of the decision: **explicit vs tacit**, eager vs lazy.
- Online learning: track realized Δ via ablations; update weights nightly (or per N runs).

- Floors: primitives/policy-pinned items can't be evicted.

6.3 Aha-merge (unification) score

`Gain = MDL_drop + Consistency_gain − Contradiction_risk − Rollback_cost`

- Cross-modal evidence allowed (vision/audio/text).
- Merges are **probationary** until downstream tests pass; each merge has a reversible receipt.

6.4 Deterministic renderer

- Inputs: graph slice + seeds + style constraints.
- Output: text/table/chart/image/audio/video artifact + render-receipt.
- Prose rendering separates **fact order** (graph-driven) from **style** (probe suggestions; never alter facts).

6.5 Round-trip metric (R^2T^2)

- For any adapter, measure `similarity(render(parse(A)), A)`; modality-specific metrics (SSIM/PSNR for images, WER/LER for audio/text, structural metrics for tables/charts).
- Store R^2T^2 in receipts; auto-promote fidelity when below policy thresholds.

---

7) Modality adapters

Parsers (input → graph)

- **Text:** sections/claims/questions/citations; resolve entities; attach provenance.
- **Table/CSV:** schema → entities/fields; constraints; units.
- **Chart:** grammar → encodings; axes; data links.
- **Image:** segmentation → parts/whole; vectors; materials; light probes (optional).
- **Audio:** spectrogram → events/tracks/phonemes; envelopes; speaker/room (policy-gated).
- **Video:** shots/scenes; objects/actions; motion fields; A/V sync edges.

Renderers (graph → output)

- Text (deterministic seeded NLG), Table (schema-first), Chart (GoG), Image (layout→vector→raster), Audio (timeline/synthesis), Video (storyboard→keyframes→export).
- All renderers produce receipts and support **round-trip** checks.

8) API (minimal)

```
POST /mosaic/plan          {goal, constraints, seeds} → {plan_id, graph_uri}
POST /mosaic/expand        {plan_id, node_id?, budget} → {graph_delta,
receipts}
POST /mosaic/render        {plan_id, target, style, seeds} → {artifact_uri,
render_receipt}
POST /mosaic/roundtrip     {artifact_uri} → {recovered_graph_uri, r2t2_score}
POST /mosaic/receipts      {plan_id} → {receipts[]}
```

---

9) UX spec (v0)

Canvas

- Center node (R) with acceptance chips.
- Expansion waves (rings); swimlanes for parallel subgoals.
- Status chips: Proposed / Amber / Verified; receipts badge.
- Coverage meter: obligations %, evidence %, open questions.

Interactions

- Ripple reveal on expansion; amber shimmer for uncertain nodes.
- Click any node → “Show Rechenweg” (deterministic step-through).
- Delta flash on edits; dependent nodes briefly highlight; preview updates in lockstep.
- Scheduler ribbon: explains **why** a node was chosen (“high VoI, low cost”).

Preview pane

- Live outline/text snapshot; deterministic; lock seeds to freeze.
- Toggle to chart/image/audio/video preview depending on target.
- HUD: tokens/latency/energy vs. baseline.

---

10) Safety, privacy, and policy

- **Safety as data:** typed goals, constraints, approvals are nodes/edges; checked before rendering.
- **Amber blocking:** contradictions and policy violations halt rendering until resolved.
- **Privacy membranes:** sensitive nodes encrypted; face/voice identity anchors require consent flags.
- **Provenance watermark:** media re-renders carry embedded provenance; receipts link edits.
- **Replayability:** any artifact can be replayed under same seeds or compared against alternative paths.

---

11) Performance & budgets

- **Budgets:** tokens, latency, energy; per-pass caps with carryover.
- **Caching:** memoize stable expansions; invalidate via provenance change.
- **Adaptive fidelity:** promote from tacit→explicit on teach/audit/fragility triggers.

---

12) KPIs (initial targets)

- Token reduction vs. strong LLM baseline: **≥60%** for comparable quality.
- Latency reduction: **≥3×** end-to-end for common tasks.
- Deterministic replay success: **~100%** under same seeds.
- R^2T^2 ≥ policy thresholds per modality (e.g., images SSIM ≥ 0.95 default).
- User trust: **≥90%** “can see why” in UX studies.
- Energy per output (J): **≥50%** reduction vs. baseline.

---

13) Roadmap

0–30 days

- Implement VoI Context Assembler (text-only) and middle-out for documents.
- Coverage meter + receipts UI; deterministic text renderer.
- A/B vs. LLM: tokens/latency/quality.

31–60 days

- Add “aha-merge” with reversible receipts; amber contradictions.
- Chart/Table adapters; R^2T^2 for text/table/chart.
- First design-partner workflow (compliance brief).

61–90 days

- Vision→graph MVP with interactive labeling; image renderer with relight/stage.
- Video storyboard renderer; audio timeline with basic VO.
- Public demo script with live replay and receipts.

---

14) Test plan

- **Unit:** selector scoring, renderer determinism, receipt hashing, merge/split reversibility.
- **Integration:** end-to-end replay; counterfactual propagation; policy gates.
- **Ablations:** with/without VoI; with/without aha-merge; tacit vs explicit.
- **Human eval:** structure coherence, factuality, UX trust, time-to-edit.
- **Performance:** tokens, latency, energy per output across standard tasks.

---

15) Risks & mitigations

- **Scheduler thrash** → add hysteresis; cap expansions/turn; cache decisions briefly.
- **Style incoherence** → style probe pass; templated micro-structures.
- **Over-merging** → probation merges, amber, automatic rollback on contradiction.
- **Adapter drift** → R^2T^2 monitoring; auto-promote fidelity or retrain.
- **Privacy** → membranes, redaction at render time, consent flags.

---

16) Provisional patent notes (enablement hints)

- **VoI context assembly:** representation-aware scoring; tiered assembly; online Δ learning via receipts.
- **Aha-merge reward:** MDL-based gain + consistency; reversible merge receipts; cross-modal

gates.

- **Middle-out generation:** acceptance-test-driven recursion; deterministic renderer separation; coverage/consistency meters.
- **Native vision/audio→graph:** factorized perception graph; active labeling; R^2T^2; relight/restage with provenance.

---

17) Open questions

- Best universal form for **invariants** across modalities?
- How to balance **late rendering** with human feedback on tone/flow?
- Policy defaults for **identity anchors** (faces/voices) across jurisdictions?
- Standard thresholds for **R^2T^2** per modality and use case?
- Which telemetry slices are safe to keep vs. privacy-sensitive by default?

---

18) Appendix — Example graph & receipts (mini)

Result R: “1-page launch brief.”

Obligations: value prop, 3 proof points, safety stance, CTA.

Expansion: for each proof point → claim, evidence binding, invariant.

Render: deterministic text paragraphs; receipts link each sentence to nodes.

```
{
  "node": {"id":"nR","type":"Result","goal":"Launch brief"},
  "edges": [
    {"from":"nR","to":"s1","type":"contains"},
    {"from":"nR","to":"c1","type":"contains"},
    {"from":"c1","to":"e1","type":"verifies"}
  ]
}
```

Receipt excerpt

{"id":"rcpt_tx1","transform":"deterministic_render","seeds":"0xabc…","artifact":"s3://…/brief_v1.txt","hash":"sha256:…"}

#ideas/phaneron