Pre-Coding Essentials (Component: crates/vm\_pipeline/src/allocate.rs, Version/FormulaID: VM-ENGINE v0) — 53/89

1. Goal & Success

Goal: Given each Unit’s UnitScores and magnitude, assign seats/power using the chosen method (WTA, D’Hondt, Sainte-Laguë, Largest Remainder) and emit UnitAllocation for downstream aggregation.

Success: Integer/rational math only; honors PR entry threshold; stable ordering; tie handling per policy; totals equal the Unit’s magnitude (or 100% for WTA).

1. Scope

In scope: Per-Unit allocation; PR threshold filtering; deterministic tie handling (and RNG path if configured).  
Out of scope: Tabulation, gates/frontier, reporting.

1. Inputs → Outputs (with schemas/IDs)

Inputs: UnitScores (natural tallies); Unit.magnitude; Params (**VM-VAR-010..012, 032..033**); option order (by order\_index).  
Output: UnitAllocation { seats\_or\_power{Option→int/%}, tie\_notes } (sums to m or 100%). Consumed by **AGGREGATE**.

1. Entities/Tables (minimal)

(Dev note: skeleton may track AllocationOutcome{allocations,tie\_logs} for audit.)

1. Variables (used here)

* **VM-VAR-010 allocation\_method** ∈ {winner\_take\_all, proportional\_favor\_big, proportional\_favor\_small, largest\_remainder, mixed\_local\_correction} *(MMP delegated elsewhere in v1)*
* **VM-VAR-011 use\_unit\_magnitudes** ∈ {on, off} *(v1: on)*
* **VM-VAR-012 pr\_entry\_threshold\_pct** ∈ integer % 0..10 *(default 0)*
* **VM-VAR-032 tie\_policy** ∈ {status\_quo, deterministic, random} *(default status\_quo)*
* **VM-VAR-033 tie\_seed** ∈ integer ≥ 0 *(default 0; used only if tie\_policy=random)*

1. Functions (signatures only)

use std::collections::BTreeMap;

use vm\_core::{

ids::{UnitId, OptionId},

entities::{UnitMeta, OptionItem},

rng::TieRng,

variables::TiePolicy,

};

use crate::tabulation::UnitScores;

pub fn allocate\_all(

unit\_scores: &BTreeMap<UnitId, UnitScores>,

units: &BTreeMap<UnitId, UnitMeta>, // includes magnitude

p: &Params,

) -> BTreeMap<UnitId, UnitAllocation>;

fn apply\_threshold(

scores: &BTreeMap<OptionId, u64>,

p: &Params

) -> BTreeMap<OptionId, u64>;

fn allocate\_wta(

scores: &BTreeMap<OptionId, u64>,

m: u32,

tie: TiePolicy,

rng: Option<&mut TieRng>

) -> UnitAllocation;

fn allocate\_dhondt(

scores: &BTreeMap<OptionId, u64>,

m: u32,

tie: TiePolicy,

rng: Option<&mut TieRng>

) -> UnitAllocation;

fn allocate\_sainte\_lague(

scores: &BTreeMap<OptionId, u64>,

m: u32,

tie: TiePolicy,

rng: Option<&mut TieRng>

) -> UnitAllocation;

fn allocate\_largest\_remainder(

scores: &BTreeMap<OptionId, u64>,

m: u32,

tie: TiePolicy,

rng: Option<&mut TieRng>

) -> UnitAllocation;

// helpers

fn break\_tie(

context: &'static str,

contenders: &[OptionId],

options: &[OptionItem],

tie: TiePolicy,

rng: Option<&mut TieRng>

) -> OptionId;

1. Algorithm Outline (per method)

**Precheck (WTA):** if allocation\_method = winner\_take\_all then assert m = 1 (also validated upstream). Winner is max Unit score; grant **100%** power. Ties per policy.

**PR threshold:** for proportional/LR, drop options with share < VM-VAR-012. Keep deterministic option order for survivors.

**D’Hondt (highest averages 1,2,3…):** iterate seat slots, choose max quotient each step; record any last-seat tie context.

**Sainte-Laguë (odd divisors 1,3,5…):** same loop with odd sequence.

**Largest Remainder:** compute exact ideal = m \* score / sum\_scores (rational); assign floors; distribute remaining seats by largest fractional remainder.

**Tie handling (common):**

* status\_quo → if SQ present among contenders, pick it; else fall back to deterministic.
* deterministic → smallest (order\_index, OptionId).
* random → uniform with **ChaCha20** seeded by **VM-VAR-033 tie\_seed**; log per pipeline TieLog rules.

**Postconditions:** Σ seats == m (or 100% WTA). Emit tie\_notes when policy applied. Allocation trail available for audit.

1. State Flow

Pipeline: **TABULATE → ALLOCATE → AGGREGATE** (fixed). UnitAllocation feeds hierarchy aggregation.

1. Determinism & Numeric Rules

Stable ordering: Options by (order\_index, id); Units by ID.  
Integer/rational comparisons only; no presentation rounding here.  
RNG only if tie\_policy = random, seeded by **VM-VAR-033**; same inputs + same seed ⇒ identical outcomes/logs.

1. Edge Cases & Failure Policy

All scores zero ⇒ tie among all options (resolve per policy).  
Threshold excludes all options ⇒ allocate zeros; downstream label/report handle.  
If a blocking last-seat tie must be logged, return typed TieContext/TieError or pass to tie stage per design.

1. Test Checklist (must pass)

* **VM-TST-001:** Sainte-Laguë with m=10 and approvals {10,20,30,40} ⇒ **1/2/3/4**.
* **VM-TST-002:** WTA with m=1, plurality {10,20,30,40} ⇒ **D gets 100%**.
* **VM-TST-003:** LR vs D’Hondt vs Sainte-Laguë with m=7, shares **34/33/33** ⇒ **3/2/2** for all three.
* Deterministic order respected (A > B > C > D); totals equal m.