# **Pre-Coding Essentials (Component: crates/vm\_pipeline/src/validate.rs, Version/FormulaID: VM-ENGINE v0) — 51/89**

## **1) Goal & Success**

Goal: Perform **structural and semantic** validation of loaded inputs **before any math**; produce a ValidationReport { pass|fail, issues[] }.

Success: On pass=false, pipeline **labels run Invalid and skips stages 3–8**; still packages Result/RunRecord with reasons.

## **2) Scope**

In scope: Checks on **hierarchy**, **magnitudes**, **ballot & tally shapes**, **WTA constraint**, **weighting data**, **quorum data**, **double-majority family preconditions**, **frontier prerequisites**. Prefer reporting issues over throwing.

Out of scope: Tabulation, allocation, gates math, frontier mapping, reporting.

## **3) Inputs → Outputs**

Input: LoadedContext (Registry+Units+Adjacency, Options with order\_index, BallotTally, ParameterSet snapshot).

Output: ValidationReport { pass|fail, issues[] } with typed severities/codes.

## **4) Entities/Tables (minimal)**

## **5) Variables (validated here)**

## **6) Functions (signatures only)**

rust

CopyEdit

pub struct ValidationIssue {

pub severity: Severity, // Error | Warning

pub code: &'static str, // e.g., "Hierarchy.TreeViolation"

pub message: String,

pub where\_: EntityRef, // Unit/Option/Tally/Param ref

}

pub struct ValidationReport { pub pass: bool, pub issues: Vec<ValidationIssue> }

pub fn validate(ctx: &LoadedContext) -> ValidationReport;

// helpers (pure, deterministic)

fn check\_hierarchy(reg: &DivisionRegistry) -> Vec<ValidationIssue>;

fn check\_magnitudes(units: &[Unit]) -> Vec<ValidationIssue>;

fn check\_ballot\_shapes(tly: &BallotTally, p: &Params) -> Vec<ValidationIssue>;

fn check\_wta\_constraint(units: &[Unit], p: &Params) -> Vec<ValidationIssue>;

fn check\_weighting(units: &[Unit], p: &Params) -> Vec<ValidationIssue>;

fn check\_quorum\_data(units: &[Unit], p: &Params, tly: &BallotTally) -> Vec<ValidationIssue>;

fn check\_double\_majority\_family(p: &Params, reg: &DivisionRegistry) -> Vec<ValidationIssue>;

fn check\_frontier\_prereqs(p: &Params, reg: &DivisionRegistry, adj: &[AdjEdge]) -> Vec<ValidationIssue>;

## **7) Algorithm Outline (checks to implement exactly)**

**Hierarchy**: Units form a **tree** (one root, no cycles). Error on violations.

**Magnitudes**: magnitude ≥ 1 for every Unit.

**Ballot & tallies**:

BallotTally.ballot\_type == VM-VAR-001.

**Tally sanity:** per Unit, Σ(valid tallies) + invalid\_or\_blank ≤ ballots\_cast.

Ranked/score datasets present/consistent if selected.

**WTA constraint**: if allocation\_method = winner\_take\_all, enforce **all Units m=1**.

**Weighting**: if weighting\_method = population\_baseline, require **positive** population\_baseline and population\_baseline\_year.

**Quorum data**: if global/per-unit quorum set, enforce presence and eligible\_roll ≥ ballots\_cast.

**Double-majority scoping**: if double\_majority=on **and** frontier=none, require family\_mode ∈ {by\_list, by\_tag} and that the resolved family is **non-empty**.

**Frontier prerequisites (shape only)**: when a frontier mode is chosen, ensure bands configured non-overlapping/ordered and adjacency edge types are valid; detailed mapping happens later.

## **8) State Flow**

LOAD → VALIDATE (fail ⇒ Invalid path) → TABULATE … (fixed order).

## **9) Determinism & Numeric Rules**

Integer/rational comparisons; **no floats**; **round half-even** only at defined decision points (none here). Stable deterministic ordering (Units by ID; Options by order\_index then ID). **Offline** only.

## **10) Edge Cases & Failure Policy**

Prefer **reporting** issues; throw only when packaging even an Invalid result is impossible (catastrophic schema contradictions).

Missing provenance or baseline years are **errors** when required by mode/weighting.

## **11) Test Checklist (must pass)**

Synthetic registries: tree passes; cycle/rootless fails with Hierarchy.TreeViolation. (Spec §ValidateInputs.)

Tally sanity vectors (per-Unit) flagged correctly.

WTA config over multi-seat units yields MethodConfigError.

Quorum data checks enforce eligible\_roll presence and bounds when quorum enabled.

When double\_majority=on & frontier=none, empty/ill-scoped family is flagged.

On pass=false, pipeline follows **Invalid path** and still builds outputs.