Rhino Credit Standard and Specification (RCSS)

Version 1.0
Author: Ngoc Hong Nguyen, Ph.D.
Affiliation: OnTeam Research Group
Date: December 2025

The Rhino Credit Standard (RCS) is a framework for measuring relative contribution in team-based work under a fixed-sum (100%) constraint.
The Rhino Credit Standard Specification (RCSS) formally specifies the rules, definitions, and computational engines required for compliance with RCS.

While prior peer-assessment systems have employed fixed-point allocation mechanisms, RCSS is, to our knowledge, the first framework to explicitly define contribution as a conserved quantity expressed in percentages, and to require preservation of this invariant across all compliant computations.

RCSS is an independent measurement standard: it defines how contribution is normalized, without reference to any specific tool, workflow, or grading scheme. It operates at the level of measurement definition rather than within the design space of peer-evaluation or grading tools. By treating contribution as a conserved quantity that exists *before* grades, money, or rewards, RCSS provides a clean formalization of contribution conservation. Existing systems, designed primarily for pedagogy rather than accounting, often violate this implicitly. RCSS does not compete with those systems; it resolves a structural inconsistency and enforces a stronger mathematical invariant, enabling consistent use of contribution data across grading, compensation, and governance contexts.

Scope and Responsibility

RCSS is a technical measurement standard. It does not prescribe organizational policy, decision-making authority, or legal compliance obligations. Responsibility for the use, interpretation, and consequences of RCSS-based implementations rests solely with the implementing and operating entities. The authors and maintainers of RCSS assume no liability for decisions made using RCSS-compliant systems.

Executive Summary

Teams produce shared outcomes, but most systems lack a consistent way to determine *who contributed how much*. Existing peer-assessment and performance tools often mix grading policy with measurement, allow totals greater than 100%, or change meaning across platforms. This creates confusion for students, instructors, managers, auditors, and HR systems.

The Rhino Credit Standard (RCS) introduces a simple governing law:

All contributions within a team must sum to 100%.

This fixed-sum principle or Ngoc's Law treats contribution as a conserved quantity, independent of grades, money, or rewards. On top of this law, RCS provides a clear and auditable method for allocating proportional credit among team members.

RCS establishes:

- a universal contribution unit **(**Rhino Credit, RC**)**
- a structured representation of team contribution (Team Contribution Vector, TCV)
- three compliant computation engines:
  - Engine A: linear normalization (baseline)
  - Engine B: Bayesian stabilization for education
  - Engine C: enterprise Bayesian network for HR contexts
- an implementation-ready data schema for software systems
- compliance levels for educational, HR, and enterprise deployment

The standard separates measurement from policy. RCS determines proportional contribution; institutions remain responsible for how contribution translates into grades, bonuses, authorship order, promotion, or compensation.

This document is Version 1.0 of the Rhino Credit Standard and Specification (RCSS). It is released as a public standard to support:

- universities and schools evaluating team projects
- companies distributing bonuses or equity
- research groups allocating authorship credit
- software developers building peer-assessment or HR tools

RCSS v1.0 is intended as a stable reference. Community feedback, independent evaluation, and future revisions are encouraged while preserving the validity of this version.

1. Introduction

Team-based work is prevalent across education, management, and organizational settings. However, existing approaches to allocating credit and responsibility within teams often lack consistent normalization and transparency. Common issues include:

- Lack of transparent contribution measurement
- Inconsistent grading or evaluation practices across teams and institutions
- Susceptibility to interpersonal bias or dominance effects
- Distortions caused by free-riding or disproportionate contribution
- Limited interoperability between evaluation systems

The Rhino Credit Standard (RCS) addresses these issues by providing:

- A single universal unit of team contribution
- A guaranteed 100% sum constraint
- Mathematically rigorous normalization of relative contribution
- Compatibility with education, HR, and performance management systems

RCS aims to play a role for contribution measurement similar to what NPS plays for customer sentiment: a simple, universal way to standardize and compare relative contribution in teams.

2. Definitions

Rhino Credit (RC)

A Rhino Credit (RC) represents an individual's proportional contribution to a completed team event, expressed as part of a fixed-sum distribution.

Team Contribution Vector (TCV)

For a team consisting of $N \in \mathbb{N}$contributors, define the Team Contribution Vector (TCV) as follows.

Let:

- $N \in \mathbb{N}$ denote the number of contributors participating in a completed team event.
- $i \in \{1,2,\ldots,N\}$ index individual contributors.

- $RC_i \in \mathbb{R}_{\geq 0}$ denote the Rhino Credit assigned to contributor i, representing that contributor's proportional share of contribution.
- $\mathrm{TCV} = (RC_1, RC_2, \dots, RC_N)$ denote the Team Contribution Vector, which aggregates the Rhino Credits of all contributors for the event.
- $\mathbb{R}^N_{\geq 0}$ denotes the non-negative real vector space of dimension N

The Team Contribution Vector is subject to the normalization constraint:

$$\sum_{i=1}^{N} R\,C_i = 100.$$

2.3 Contribution Event (CE)

A Contribution Event (CE) is a discrete, completed team task for which individual contributions are evaluated.

2.4 Raw Rating Matrix (RRM)

For a team of N contributors, define the Raw Rating Matrix $\mathrm{R} \in \mathbb{R}^{N \times N}$ as:

$$\mathbf{R} = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1N} \\ R_{21} & R_{22} & \cdots & R_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ R_{N1} & R_{N2} & \cdots & R_{NN} \end{pmatrix}$$

where R_{ij} denotes the rating assigned by contributor i to contributor j, and diagonal entries R_{ii} represent self-ratings.

3. Design Principles

RCS is governed by the following normative constraints:

1. 100% Law or Ngoc's Law**:** The sum of all contributions within a team must equal 100%.
2. Relative Contribution: Rhino Credits measure proportional contribution, not absolute effort, time, or hours worked.
3. Non-negativity**:** Rhino Credits must be non-negative ($RC_i \geq 0$).
4. Interoperability**:** The standard is implementation-agnostic and may be integrated across diverse software systems.

5. Auditability: All computations must be reproducible from recorded inputs and declared normalization procedures.

4. Computational Engines

This section specifies the canonical computation methods for Rhino Credits (RC), as defined in Section 3.

RCSS defines three compliant computation engines (A, B, C) to accommodate differing data availability and governance requirements, while preserving a single conserved contribution unit and the 100% normalization law.

4.1 Engine A: Linear Normalization (Baseline Standard)

Engine A is a linear normalization engine that converts arbitrary contribution signals into a conserved contribution allocation summing to 100%, preserving relative rankings while eliminating inflation, scale distortion, and grade overshoot.

Applicability
Engine A is the baseline RCSS-compliant computation method. It is intended for lightweight implementations and environments with limited data availability.

Description
Engine A computes Rhino Credits (RC) by applying linear normalization to mean contribution ratings. It is fully deterministic and does not incorporate probabilistic or Bayesian inference.

Input rating scale (mandatory)

All entries of the Raw Rating Matrix (RRM) SHALL satisfy:

- integer values only
- closed interval 0–100
- decimals NOT permitted at rating stage

Formally:

$$R_{ij} \in \{0,1,2,\ldots,100\}$$

These values represent: the percentage of total team contribution that rater i believes contributor j provided.

This is not a Likert scale, rubric score, or ordinal category. It is a direct percentage claim.

100% Law (Ngoc's Law)

For any valid RCSS computation:

$$\sum_{j=1}^{N} R\,C_j = 100$$

and for all contributors:

$$RC_j \geq 0$$

For each contributor j, compute the mean received rating M_j from the Raw Rating Matrix (RRM):

Step 1: Compute Mean Received Rating

For each contributor j, compute the arithmetic mean rating received:

$$M_j = \frac{1}{N}\sum_{i=1}^{N} R_{ij}$$

where:

- R_{ij}= percentage assigned by rater $i$to contributor j
- N= number of team members

Interpretation:

$M_j$is the central estimate of perceived contribution share before normalization.

Step 2: Linear normalization

Rhino Credits RC_j shall be computed as:

$$RC_j = \frac{M_j}{\sum_{k=1}^{N} M_k} \times 100$$

This guarantees total Rhino Credits equal 100%, values remain non-negative, and results are scale-invariant (uniform inflation cancels out).

Properties of Engine A

Engine A:

- enforces the 100% conservation law
- outputs proportional contribution shares
- prevents grade overshoot beyond 100%
- is deterministic and reproducible
- contains no Bayesian or probabilistic inference
- degrades gracefully under uniform inflation
- preserves rank order of M_j

Engine A does NOT:

- evaluate effort
- judge fairness
- apply penalties
- detect bias
- reinterpret ratings

It simply translates percentage claims to conserved contribution shares.

4.2 Engine B: Bayesian Normalization (Academic Standard)

Engine B extends Engine A to address uncertainty, noise, and small-sample effects commonly present in educational peer-rating data.

Engine B does not replace Engine A. It refines contribution signals prior to Engine A's mandatory normalization step.

This separation is a core RCSS design constraint.

Design requirements (normative)

An Engine B–compliant implementation shall:

- Preserve the 100% Law
- Reduce sensitivity to noisy or sparse peer ratings
- Treat all contributors symmetrically in the absence of evidence
- Remain interpretable to instructors and auditors
- Converge to Engine A behavior as evidence accumulates

Any Bayesian construction that violates the 100% Law is non-compliant, regardless of sophistication.

Input scale rule (no Likert)

Engine B uses the same input constraint as Engine A:

- integer values only
- 0–100 inclusive
- no decimals at the rating stage
- direct percentage contribution claims

Formally:

$$Rij \in \{0,1,2,\ldots,100\}$$

These values represent the percentage of total team contribution that rater (i) believes contributor (j) provided.

Engine B does not allow Likert scales, ordinal rubrics, or agreement categories.

Conceptual model

Peer ratings in educational settings are typically:

- sparse
- socially biased
- unevenly calibrated
- low-signal at small sample sizes

Accordingly, Engine B treats each contributor's underlying contribution as **latent**; not directly observed, but inferred from percentage signals.

Let:

- R_{ij}= percentage rating given by rater $i$to contributor j
- $S_j = \sum_i R_{ij}$= aggregated received percentage signal
- θ_j= latent contribution strength of contributor j

The latent variable $\theta_j$represents the true contribution of a participant, which is not directly observable and is inferred only indirectly through peer signals.

Prior (neutral baseline)

Engine B applies an identical, weakly informative prior to all contributors:

$$\theta_j \sim \text{Gamma}(\alpha_0, \beta_0)$$

RCSS canonical educational default:

- $\alpha_0 = 1$
- $\beta_0 = 1$

Interpretation:

Before observing ratings, all contributors are assumed equal but uncertain. The prior is symmetric and minimally informative.

In Engine B, parameters α and β represent neutral Bayesian stabilizers corresponding to one virtual observation per contributor and one virtual rater, respectively. The canonical RCSS default (α = 1, β = 1) provides the minimal non-arbitrary prior that preserves symmetry, prevents overreaction to noisy peer ratings, and converges rapidly to data-dominated estimates as evidence accumulates. Alternative parameterizations may be used for domain-specific policies, but must be explicitly declared, as deviations alter the sensitivity of the system rather than the conservation constraint imposed by the 100% Law.

α and β are neutral Bayesian stabilizers that prevent overreaction to noisy peer ratings while preserving proportional contribution under the 100% Law.

θ_j represents the inferred contribution strength of contributor j, estimated from peer signals under Bayesian stabilization, prior to normalization under the 100% Law.

Before observing peer ratings, all contributors are assumed equal but uncertain.

The Gamma distribution is used because it is:

- strictly non-negative
- interpretable as contribution mass
- conjugate to additive signals
- stable under aggregation

Likelihood (aggregation of peer signal)

The aggregated peer signal S_j is modeled as conditionally proportional to latent contribution strength:

$$S_j \mid \theta_j \sim \text{Poisson}(\lambda_j), \lambda_j \propto \theta_j$$

This formulation does not claim that peer ratings follow a Poisson process in reality. The Poisson–Gamma construction is used purely as a mathematical aggregation mechanism that allows noisy contribution signals to be combined and smoothed using a transparent, closed-form expression.

Posterior inference

Given conjugacy:

$$\theta_j \mid S_j \sim \text{Gamma}(\alpha_0 + S_j, \beta_0 + N)$$

The notation "|" denotes conditioning on observed data; in Engine B it represents the update of contribution estimates after observing peer signals.

The posterior mean is:

$$\widetilde{M}_j \equiv \mathbb{E}[\theta_j \mid S_j] = \frac{\alpha_0 + S_j}{\beta_0 + N}$$

$\widetilde{M}_j$ is the Bayesian-adjusted contribution signal. This Poisson–Gamma construction is used as a minimal conjugate device to stabilize additive peer signals; no behavioral or psychological interpretation is implied.

Key properties:

- Shrinks extreme values toward equality under weak data
- Dampens single-rater dominance
- Never produces negative values
- Converges to linear aggregation as data grows

The posterior mean combines observed peer signal with a neutral baseline, producing a stabilized contribution estimate that converges to linear normalization as data increases

Mandatory normalization

Engine B shall not allocate Rhino Credits directly. Bayesian methods are used only to stabilize noisy ratings, not to decide final shares. Final allocation is always performed by Engine A so that total contribution remains exactly 100%.

All Bayesian-adjusted signals must be passed to Engine A for normalization:

$$RC_j = \frac{\widetilde{M}_j}{\sum_{k=1}^{N} \widetilde{M}_k} \times 100$$

Compliance with RCSS axioms

Normalization mathematically enforces:

$$\sum_{j=1}^{N} RC_j = 100$$

This conservation constraint guarantees that Rhino Credits represent a zero-sum allocation of contribution within a team, rather than an independent scoring or grading system. Bayesian inference may modify intermediate estimates, but the final normalization step mathematically enforces the invariant that total contribution remains exactly 100%.

Engine B has backward compatibility

As peer evidence accumulates, Engine B converges to Engine A behavior.

$$\tilde{M}_j \to \frac{S_j}{N} \quad \Rightarrow \quad RC_j \to \text{Engine A result}$$

4.3 Engine C: Enterprise Bayesian Network (HR/ Compensation Standard)

Engine C estimates each contributor's latent contribution strength using all available evidence, then converts those estimates into conserved Rhino Credits through mandatory Engine A normalization.

Purpose and Scope

Engine C defines the enterprise-grade inference engine for Rhino Credit allocation in high-stakes environments, including:

- bonus pools
- performance-based compensation
- promotion review
- equity or profit-sharing decisions
- regulated or audited HR systems

Engine C extends Engine B by incorporating multi-layer Bayesian inference, while remaining fully compliant with RCSS invariants, including the 100% Law and mandatory Engine A normalization.

Engine C is designed for contexts where:

- financial consequences are material,
- auditability is mandatory,
- historical data is available,
- and governance requirements exceed those of education.

Engine C extends Engine B but does not alter the Rhino Credit unit, the 100% Law, or the normalization requirement**.**

Engine C does not replace Engine A or Engine B.

Instead:

- Engine A defines the allocation law
- Engine B defines single-period stabilized inference

- Engine C generalizes inference across raters, time, and organizational context

Engine C refines contribution estimates only.
Final allocation is always enforced via Engine A normalization.

Input scale rule (mandatory. No Likert)

Engine C accepts the same inputs as Engines A and B:

- integer values only
- 0–100 inclusive
- no decimals at rating stage
- direct percentage contribution claims

Formally:

$$Rij \in \{0,1,2,\ldots,100\}$$

These values represent the percentage of team output rater (i) believes contributor (j) provided.

Engine C does not accept Likert scales, ordinal rubrics, or agreement categories. All inputs must be direct percentage estimates.

Conceptual Model

In enterprise settings, peer ratings are not independent or equally reliable.

Engine C therefore models contribution inference as a Bayesian network, incorporating:

- contributor latent strength
- rater reliability
- self-assessment bias
- historical performance signals
- temporal stability

Let:

- θ_j= latent contribution strength of contributor j
- RRM = Raw Rating Matrix
- $\mathcal{B}$= rater bias and reliability profiles
- $\mathcal{H}$= historical contribution data

The goal of Engine C is to infer $\theta_j$using all available evidence, subject to governance constraints.

Enterprise Inference Components

An RCSS-compliant Engine C may include, but is not limited to, the following modules.

Rater Reliability Weighting

Rater reliability weighting scales the influence of rating signals based on observed consistency and stability over time, without excluding any rater or altering the conservation constraint. Each rater i may be assigned a reliability weight w_i, inferred from:

- historical agreement with group consensus
- variance patterns across evaluations
- stability over time

Unreliable raters contribute less influence, without being excluded.

Self-Assessment Bias Modeling

Self-assessment ratings are treated as a distinct signal channel due to their asymmetric incentive structure and predictable bias characteristics. RCSS-compliant systems may:

- down-weight self-ratings
- model them using explicit inflation priors
- exclude them from inference while retaining them for audit

The chosen treatment MUST be explicitly declared.

Consistency and Residual Analysis

Engine C may perform consistency and residual analysis to assess the coherence and stability of rating signals, including internal rater inconsistency, deviations from expected contribution patterns, and anomalous scoring behavior. Such analysis may inform inference confidence, rater reliability weighting, or audit flags, but MUST NOT override normalization or violate the 100% Law.

Time-Series Contribution Priors

Where historical contribution data exists, Engine C may incorporate time-series contribution priors to improve stability across repeated evaluations. Such priors may reflect past contribution distributions, decay-weighted performance, and role-adjusted expectations. Temporal priors must be contributor-specific, time-stamped, and bounded to prevent lock-in or reputational entrenchment, ensuring that current evidence remains the dominant factor in inference.

Posterior Inference

Engine C computes the posterior expectation of latent contribution strength using all available evidence in the Bayesian network.

$$\mathbb{E}[\theta_j \mid \mathrm{RRM}, \mathcal{B}, \mathcal{H}]$$

Engine C computes the posterior expectation of each contributor's latent contribution strength based on all available evidence. The inference machinery may be arbitrarily complex, provided that all priors are explicit, assumptions are documented, and results are reproducible. The output of inference is an unconstrained contribution signal and shall not be interpreted as a final allocation; normalization under Engine A is mandatory.

Mandatory Normalization (RCSS Invariant)

As with Engines A and B, allocation is performed only via normalization:

$$RC_j = \frac{\mathbb{E}[\theta_j \mid \mathrm{RRM}, \mathcal{B}, \mathcal{H}]}{\sum_{k=1}^{N} \mathbb{E}[\theta_k \mid \mathrm{RRM}, \mathcal{B}, \mathcal{H}]} \times 100$$

As with all RCSS-compliant engines, final allocation shall be performed exclusively through normalization of inferred contribution signals to a total of 100%. This step is mandatory and non-negotiable; any implementation that bypasses or alters normalization is non-compliant.

Auditability and Governance Requirements

Engine C implementations must support full auditability and governance by logging all priors, hyperparameters, inference model versions, raw rating data, and normalization results. These records must enable post-hoc reconstruction of any allocation outcome. In regulated environments, logs should be append-only, tamper-evident, and subject to explicit data-retention controls.

What Engine C Explicitly Does NOT Do

Engine C is a measurement and analytical component only. It does not, and is not intended to:

- supersede, modify, or override any organizational, institutional, contractual, or legal policy;
- determine, recommend, or enforce salary bands, compensation levels, bonuses, promotions, grades, or employment outcomes;
- replace, constrain, or automate managerial or administrative judgment;
- assign, mandate, or execute rewards, penalties, or decisions beyond the calculation of normalized contribution shares.

Engine C produces advisory contribution metrics under a 100% fixed-sum normalization framework. All interpretations, decisions, and actions taken on the basis of these metrics remain solely the responsibility of the relevant human decision-makers and governing bodies.

Relationship to Engines A and B

- Engine A defines the canonical allocation mechanism
- Engine B stabilizes inference under limited data
- Engine C generalizes inference under governance constraints

All three engines:

- produce the same unit (Rhino Credit)

- obey the same conservation law
- interoperate within the same ecosystem

Formal Compliance Statement

An implementation is RCSS Engine C compliant if and only if it infers latent contribution strength using an explicit, auditable probabilistic model and applies Engine A normalization as the final allocation step.

Table1 outlines the three RCSS compliance levels and their minimum requirements for Engines A, B, and C. It shows a progression from basic 100%-law enforcement to audited Bayesian adjustment and, finally, to fully governed enterprise deployment.

Table 1:

Compliance Level	**Engine**	**Intended Context**	**Minimum Requirements**
RC-Compliant (Basic)	A	Lightweight and educational use	Enforcement of the 100% Law; non-negative RCs; valid Team Contribution Vector (TCV) output
RC-Certified (Academic)	B	Institutional and academic environments	Explicit prior assumptions; reproducible Bayesian adjustment; audit logs of inputs and normalization
RC-Enterprise (Reference)	C	High-stakes organizational contexts	Explicit inference assumptions; full auditability; governance and traceability controls

5. Rounding Standard

To ensure cross-system consistency and exact conservation, RCSS specifies the following rounding conventions:

- Internal precision: RC values SHOULD be stored with at least 6 decimal places.
- Display precision: RC values MAY be displayed using 1 or 2 decimal places.
- Conservation correction: If rounding results in a displayed total different from exactly 100.0, a minimal corrective adjustment SHALL be applied to one RC value to restore exact conservation.

The correction MUST preserve non-negativity and SHOULD be applied in a deterministic manner (e.g., to the largest RC value).

Example Output Format (Informative)

The following JSON structure is an illustrative example of an RCSS-compliant output. RCSS does not mandate a specific serialization format.

```
{
  “teamID”: “T2025-01”,
  “engineUsed”: “B”,
  “members”: [
    { “id”: “A”, “rc”: 38.0 },
    { “id”: “B”, “rc”: 33.8 },
    { “id”: “C”, “rc”: 28.2 }
  ],
  “timestamp”: “2025-12-12T08:00:00Z”
}
```

6. Governance **and Versioning**

The Rhino Credit Standard Specification (RCSS) is maintained as an open reference standard. Stewardship of the specification text, including clarification and version updates, is coordinated through a designated maintainer group referred to as the Rhino Standards Council (RSC).

RCSS versioning follows semantic versioning principles:

- **1.x** — backward-compatible clarifications and refinements
- **2.x** — extensions introducing new capabilities or optional components
- **3.x** — advanced reference profiles intended for high-stakes organizational contexts

Governance describes specification maintenance only and does not imply regulatory authority, certification, or enforcement.

7. Conclusion

The Rhino Credit Standard Specification (RCSS) defines a clear framework for representing proportional contribution within teams under a fixed-sum constraint (Ngoc's Law). The standard formalizes terminology, computation engines, and output requirements so that different systems can produce compatible contribution allocations.

RCSS does not prescribe grading policies, compensation decisions, or organizational governance. It provides a measurement layer that other systems may adopt or extend.

Version 1.0 is intended as a stable reference. Feedback, independent evaluation, and further refinement are explicitly welcomed for future versions of the specification.