Distributional Contracts for Stochastic Systems

Contents

1. Document History

This is a working document: a formal declaration of what the mavai family of frameworks is for and how its statistical claims are constructed, expected to evolve as the methodology matures.

#	Date	Milestone
1	**2026-01**	**First issue.** A uni-dimensional distributional contract covering only functional stochasticity: a Bernoulli-trial model over a quality predicate Q, aggregated binomially, with the Wilson-score lower bound as the mechanism for threshold derivation from baselines. Hosted alongside the punit framework.
2	**2026-03**	**Temporal dimension added.** The contract was extended from one dimension to two (functional and temporal). Latency was introduced as a non-parametric problem via empirical percentiles, with a first-generation threshold derivation using the standard error of the mean as a proxy for percentile uncertainty, $\widehat{\tau}_j = Q(p_j) + z_\alpha \cdot s/\sqrt{n_s}$.
3	**2026-04**	**Stricter latency treatment.** The latency population was formally decomposed into a tripartite contract (correctness / availability / latency-given-success), with the perverse-incentive hazard of conditioning on success named explicitly. The $s/\sqrt{n_s}$ approximation — which understated tail-percentile uncertainty for heavy-tailed distributions — was replaced by the exact binomial order-statistic upper confidence bound on the baseline quantile, restoring statistical symmetry with the Wilson-based construction on the pass-rate side. **Terminology tightening.** The term system was replaced by service throughout the document wherever the specific artefact under evaluation was meant, reserving system for generic-class and idiomatic usage. This is a subtle but load-bearing distinction: under the earlier terminology the contrast between ”the system has changed” and ”the operating conditions have changed” could collapse, because system is frequently read to subsume its environment.

#	Date	Milestone
4	**2026-05**	**Criterion decomposition of the functional dimension.** The functional dimension was refined from a single quality predicate Q into a conjunction of separately-contractual criteria, each its own Bernoulli stream with its own threshold, confidence level, and feasibility gate; the single-predicate model is recovered as the one-criterion ($m = 1$) special case. A union-bound hiding result establishes why an aggregated conjunction rate cannot be the unit of inference: rare-but-consequential per-criterion failures are absorbed into the noise of more frequent ones, becoming statistically invisible. The composite verdict was correspondingly lifted from a flat boolean to a structured tuple over per-criterion verdicts. **Clause forms** were introduced: rate-bounded clauses, discharged statistically — their thresholds normative or empirical in origin, a separate axis — and categorical (zero-failures) clauses, discharged architecturally. The term empirical was confined to the experiment-derived threshold origin and no longer names the rate-bounded form. The per-trial outcome model was reworked so that every trial counts toward its criterion's denominator, with a non-evaluable response recorded as a failure carrying a diagnostic reason rather than excluded.

The first three milestones strictly extend one another in the scope of what the contract claims, and none supersedes the Bernoulli/Wilson foundation laid in Milestone 1. Milestone 4 likewise preserves that foundation — the one-criterion case reproduces the earlier model unchanged — but it does supersede one earlier construction: the Milestone-1/3 model swept every no-testable-value outcome into a single excluded catastrophic bucket, leaving the quality predicate undefined on it. That bucket is now split — a service that produced no usable value (timeout, malformed output, refusal, or a failed transform step) is a failure carrying a diagnostic reason, while a true catastrophe (the testing machinery itself dying, e.g. `OutOfMemoryError`) remains outside the sample and aborts the run. That supersession is called out explicitly where it occurs.

2. Abstract

In stochastic systems, correctness cannot in general be characterised solely by the outcome of a single execution, since individual executions may legitimately vary even when the system is behaving acceptably overall.

This paper extends Bertrand Meyer's Design by Contract [1][2] by lifting postconditions from Boolean predicates over individual executions to statistical assertions over repeated executions. We introduce the notion of a distributional contract, under which the quality of a stochastic service is characterised along two orthogonal dimensions: functional stochasticity — whether the service produces acceptable results with sufficient probability — and temporal stochasticity — whether it responds

within acceptable time bounds with sufficient probability. These dimensions require distinct statistical treatments but together address the two most fundamental quality concerns for a stochastic service.

Within the functional dimension, the single notion of an acceptable result is itself refined into a conjunction of separately-checkable criteria, each assessed as its own statistical stream rather than folded into one aggregate rate. The obligations a contract carries are further partitioned by form into rate-bounded clauses, whose tolerable failure rate is bounded statistically, and categorical clauses, whose intolerable failures are bounded architecturally — the rate-bounded threshold being, on a separate axis, either normative or empirical in origin.

Since the quantities of interest are not directly observable, they are estimated empirically from repeated executions and assessed using conservative statistical bounds. This yields an operational basis for making statistically grounded assertions, at a defined confidence level, that a stochastic service satisfies its contractual obligations.

The statistical foundations on which these methods rest — the Wilson score construction, conservative threshold derivation, the feasibility constraint, the per-criterion model, and non-parametric latency inference — are developed and justified in full in the Statistical Companion [3], the canonical methodology document of the mavai family. The present paper builds on that work: it supplies the contractual framing and describes each method to the depth needed to convey the model, deferring the formal statistical justification to the companion.

3. Two Dimensions of Stochasticity

In deterministic software, correctness is defined pointwise: for any input satisfying the precondition, the postcondition must hold for the resulting output. This model is insufficient for stochastic systems, where individual executions may legitimately vary even when the system is behaving acceptably overall.

The idea of assessing stochastic systems through repeated execution and statistical inference has precedent in the tradition of statistical model checking, where Younes and Simmons [4] showed that time-bounded probabilistic properties can be assessed by hypothesis testing over sampled executions, trading absolute certainty for bounded decision error, and Legay, Delahaye, and Bensalem [5] surveyed this approach as a scalable alternative to numerical model checking. We extend this line of reasoning from formal stochastic models to operational software services, and from temporal-logic property satisfaction to a contractual framing: the notion of a contract is lifted from individual executions to distributions over executions.

The contract idea has been lifted before. Benveniste et al. [6] generalised it from Meyer's routine-level preconditions, postconditions, and invariants to a meta-theory of contract-based system design, in which a component's guarantees hold relative to explicit assumptions about its environment and contracts support refinement and composition. The present work makes a complementary move along the same axis, but in a statistical rather than a model-theoretic register: it lifts the contract from deterministic, compositional obligations to operational, black-box obligations over a stochastic service, where the environmental assumptions become the sampling and

covariate conditions under which evidence is drawn, and satisfaction becomes a statistically qualified verdict over repeated executions rather than a pointwise or purely compositional one.

The uncertainty exhibited by stochastic systems manifests along two orthogonal dimensions, each giving rise to a distinct form of distributional contract.

The first is **functional stochasticity**: whether the service produces an acceptable result. Given identical input, a stochastic service may produce correct output on some executions and incorrect output on others. The correctness of the output is a random variable. The distributional contract over this dimension asserts that the service succeeds with probability at least some minimum acceptable level.

The second is **temporal stochasticity**: how long the service takes to respond. Even among successful executions, response times vary substantially. A service that typically responds in 200 milliseconds may occasionally take several seconds. Latency is not a fixed property of the service; it is a distribution. The distributional contract over this dimension asserts that a given percentile of the response time distribution falls below an acceptable bound — for example, that the 95th percentile latency does not exceed 500 milliseconds.

These dimensions are orthogonal contractual concerns, not probabilistically independent random variables. A fast response can be incorrect; a slow response can be correct; a service can satisfy its functional contract while routinely breaching its latency contract, or vice versa. Neither concern subsumes the other, and a distributional contract that addresses only one leaves the other unexamined. The two may in fact be correlated in practice — timeouts are correctness failures, refusal paths may be fast, slow-success tail mass can be traded into failures — so the methodology assumes no statistical independence between them. It evaluates each concern separately and composes their verdicts by conjunction; the composite is a logical conjunction, not a probabilistic-independence model.

Crucially, the two dimensions require different statistical treatments. Functional stochasticity reduces, in the evaluable case, to a binary outcome for each execution — the quality predicate either holds or it does not — and is naturally modelled as a Bernoulli process. Not all executions are evaluable, a distinction addressed shortly. Temporal stochasticity, by contrast, is a continuous quantity; each execution yields a duration, and the contractual assertion is over the shape of the resulting distribution rather than over a binary success rate. The sections that follow develop the Bernoulli framework for functional stochasticity in detail. The statistical treatment of temporal stochasticity, which rests on empirical percentile analysis rather than binomial inference, is addressed separately.

A further refinement applies within the functional dimension. The single quality predicate above is, in all but the simplest contracts, a conjunction of distinct expectations — that a response parses, that required fields are present, that its content is acceptable to a reader, that it leaks no prohibited material. This paper treats each such expectation as a separate **criterion**, evaluated as its own Bernoulli stream with its own threshold and confidence level, rather than collapsing them into a single composite predicate. The criterion decomposition refines the evidence carried within the functional dimension; it does not introduce a third dimension of stochasticity, since

every criterion shares the same binomial machinery. The development below first establishes the per-criterion model, then sets out why aggregation into a single rate is inadequate, and finally how per-criterion verdicts compose into the contract's verdict.

4. Functional Stochasticity

In deterministic Design by Contract, a postcondition is a Boolean predicate that must hold on every execution. The functional contract for a stochastic service lifts this requirement: the obligation is not that every execution satisfies a postcondition, but that it is satisfied with at least a minimum acceptable probability. A contract is rarely a single such proposition, however — it is a conjunction of separately-checkable expectations — and the machinery below is built to keep those expectations apart rather than fuse them into one.

4.1 Postconditions, Criteria, and Samplings

The functional dimension rests on three primitives, each with exactly one job.

A **postcondition** is a named predicate over the output of a single execution: it decides, for one response, whether one observable property holds. Response parses as JSON, required fields are present, the answer is readable by a layperson, the advice does not suggest self-harm — each is a postcondition. It carries no threshold and no statistical configuration of its own. This is exactly Meyer's postcondition, with one change: it is not expected to hold on every execution, but its satisfaction on each execution is observed and recorded.

A **criterion** is the unit of statistical evaluation. It hosts one or more postconditions and, on each execution, yields a single pass/fail outcome — the conjunction of its hosted postconditions' verdicts. A criterion declares the threshold it is judged against, its confidence level, and the mode under which it is evaluated. The contractual obligation is stated per criterion: where a single-predicate contract asserts $p = \mathbb{P}(Q = 1) \geq p_{\min}$, a criterion c asserts

$$p_c \;=\; \mathbb{P}(\text{criterion } c \text{ holds}) \;\geq\; p_c^*,$$

with its own minimum acceptable rate p_c^* and its own confidence level. Two postconditions whose failures carry materially different consequences belong in two different criteria, never merged into one stream — the reason is developed shortly.

A **sampling** is the list of $N \geq 1$ inputs that an experiment or test posts to the service. The sampling is shared across every criterion the contract exercises in a single run: one list of inputs produces one response per input, and every criterion is evaluated against every response. A contract that needs evidence about a different input distribution runs a separate experiment with its own sampling.

4.2 The Per-Criterion Bernoulli Model

Each criterion c defines its own Bernoulli stream. On each execution i of the sampling it produces an indicator $X_{i,c} \in \{0, 1\}$, modelled exactly as the single-criterion trial:

$$X_{i,c} \sim \text{Bernoulli}(p_c), \qquad K_c = \sum_{i=1}^{n_c} X_{i,c} \sim \text{Binomial}(n_c, p_c), \qquad \hat{p}_c = K_c/n_c,$$

where n_c is the number of executions in the run — the size N of the sampling, since every sample is presented to every criterion. Since p_c is not directly observable it is estimated from such a stream, and the methodology runs two of them. An **experiment** exercises the criterion over n_{exp} trials and reports the baseline estimate $\hat{p}_{c,\text{exp}}$; this baseline is not itself the bar a later run must clear.

A subsequent **probabilistic test** exercises the same criterion over its own sampling — of size n_{test}, typically much smaller than n_{exp} — and re-estimates the rate as $\hat{p}_{c,\text{test}}$. The Wilson lower bound that decides the test is computed from the baseline estimate $\hat{p}_{c,\text{exp}}$, the **test** sample size n_{test}, and the confidence level α_c (the same α_c in experiment and test) — the test's sample size, not the experiment's — and $\hat{p}_{c,\text{test}}$ is compared against that bound. If it exceeds the bound, there is insufficient evidence to reject the null of no degradation and the criterion passes. This is the model only; the deeper exposition — the Wilson construction itself, the integer pass-cutoff that is the binding decision artefact, and the discreteness corrections — is given in the Statistical Companion §3.4.

This constraint — that the service is an opaque executable whose properties must be inferred from observed behaviour rather than from an inspectable internal model — is shared with the black-box probabilistic verification tradition. Sen, Viswanathan, and Agha [7] consider statistical model checking of black-box probabilistic systems where properties are inferred from observed traces, and Aichernig and Tappler [8] extend this to stochastic systems with probabilistic behaviour. The present work inherits their black-box constraint but reframes the objective: rather than establishing reachability or property satisfaction within automata-based frameworks, the goal is to assess whether the service meets a contractual quality obligation.

A contract with a single criterion is simply the $m = 1$ case of this model, in which the per-criterion stream coincides with the single Bernoulli model. **From here on, the single-criterion notation X_i, $\hat{p}$, p_{min}, α used elsewhere in this paper should be read as the per-criterion $X_{i,c}$, $\hat{p}_c$, p_c^*, α_c for any criterion c a contract declares: the Wilson construction, the threshold-derivation pipeline, the feasibility gate, and the verdict rule all apply unchanged to each stream.** In this way Design by Contract is generalised from pointwise correctness to distributional correctness, criterion by criterion. The constructs particular to the multi-criterion case — how the streams compose into one verdict, and the resulting false-alarm budget — are taken up after the threshold machinery.

The independence and stationarity assumptions apply per criterion: within a criterion, the per-execution indicators are treated as i.i.d. under the input distribution the sampling represents. Across criteria the methodology assumes no independence — an execution that returns malformed output may fail several criteria at once — and the consequence of that dependence for the composite verdict is disclosed rather than assumed away.

4.3 The Operational Outcome of a Trial

In operational settings the raw outcome of an execution is not a clean pass/fail. A criterion may yield an acceptable value; it may yield a value that is present but unacceptable; or it may yield no testable value at all — the service timed out, returned malformed output, omitted required material, or refused, or a transform step the criterion applies to the response failed to convert it. Earlier versions of this paper swept every no-testable-value outcome into a single bucket and excluded the whole bucket from the analysis, leaving the quality predicate undefined on it.

That bucket is now split, and the exclusion withdrawn for the part that belongs in the sample. A value the service failed to make usable is an expected, in-band outcome of a stochastic service: it counts as a failure of the criterion, not as a non-event. Excluding such outcomes from the denominator would let a service flatter its measured rate by failing to respond, and would hide a structurally unhealthy service behind a healthy-looking pass rate. The genuinely exceptional case — the test harness itself dying — is a different matter entirely, treated at the end of this section.

A criterion is evaluated on every execution of the sampling, and on each it yields exactly one of two outcomes:

- **PASS** — the criterion produced a value to test and its postconditions hold.
- **FAIL** — anything else. A FAIL carries a diagnostic **reason** that does not change the arithmetic: condition — the postcondition was evaluated and did not hold; or transform / no-value — no value to test could be produced, because the service returned no usable output (timeout, malformed response, refusal, omitted material) or a transform step the criterion applies failed to convert the response. A transform/no-value FAIL is subtly different from a condition FAIL — the postcondition was never reached — but it is an ordinary failure of the criterion, not a catastrophe.

Every execution counts toward the criterion's denominator n_c, which is therefore simply the sampling size N: every sample is seen by every criterion, and none is filtered out. The success count K_c is the number that PASS, and $\hat{p}_c = K_c/n_c$. A trial that produced no testable value is a FAIL like any other — its reason records that it was not the postcondition that failed, but the trial is neither excluded from the denominator nor counted as a success. The one place a denominator legitimately narrows is the latency dimension, whose statistics are conditional on success (see Temporal Stochasticity): if 98 of 100 trials pass, latency is summarised over those 98, whereas a functional criterion's denominator remains the full sampling. The accounting of per-trial outcomes and their reasons is detailed in the Statistical Companion (§1.4.5a).

This accounting makes a quality criterion's rate **end-to-end**: because a non-response already fails any criterion that needed the response, unavailability is reflected with no special handling. A contract that wants availability as a distinct, visible metric simply declares it as its own criterion — for example a zero-failures or rate criterion over the service returned usable output — standing alongside the others rather than altering their denominators. Availability is therefore ordinary structural composition, not a denominator mechanism, and the earlier treatment of it as an orthogonal subcontract is superseded accordingly.

One outcome remains genuinely outside this accounting: a true **catastrophe** — not a failure of the service but a failure of the testing machinery itself, such as an `OutOfMemoryError`, a `StackOverflowError`, or process termination, after which the run cannot continue in any meaningful way. A catastrophe of this kind aborts the run and is investigated as a defect; it is not a sample, and it contributes to no denominator. This is categorically distinct from the transform/no-value failures above: those are expected, in-band outcomes of a functioning test against a stochastic service, and are counted as criterion failures, whereas a catastrophe means the experiment could not be carried out at all. A transform step that fails to convert a response is the former, not the latter — the harness is alive and the criterion simply could not produce a value to test.

The latency dimension remains conditional on success — latency is measured over executions that satisfied the contract's correctness criteria, for the reasons developed under Temporal Stochasticity. Functional criteria, availability-as-a-criterion, and latency-given-success are evaluated jointly and by logical conjunction, so a service cannot trade correctness for latency under the composite verdict.

4.4 Why a Single Aggregate Rate Is Inadequate

Consider aggregating the per-criterion indicators into a single conjunction rate — one stream, one p, one Wilson interval. A union-bound argument shows what such an aggregate conceals. Let $\{C_1, \dots, C_m\}$ be the contract's criteria with per-trial indicators $X_{i,c}$, and define the conjunction indicator $X_i = \prod_{c=1}^{m} X_{i,c}$ with $p = \mathbb{P}(X_i = 1)$ — the rate at which every criterion holds simultaneously on a trial. By the union bound applied to the complementary events,

$$1 - p \;=\; \mathbb{P}\big(\exists\, c : X_{i,c} = 0\big) \;\le\; \sum_{c=1}^{m} (1 - p_c),$$

with equality if and only if the per-criterion failure events are disjoint. The aggregate failure rate is thus bounded above by the sum of per-criterion failure rates and, in typical cases, dominated by the largest of them.

The consequence is a masking effect. If one criterion c^* fails far more rarely than the others — $1 - p_{c^*} \ll \max_{c \neq c^*}(1 - p_c)$ — then $1 - p$ is essentially insensitive to $1 - p_{c^*}$: a change in the rare criterion's failure rate of any magnitude smaller than the dominant criterion's sampling noise leaves the aggregate unmoved. The conjunction rate cannot detect movement in the rare criterion. The masking is purely a property of frequency — rare failures are absorbed by frequent ones regardless of what they mean — and that is precisely the hazard, because high-consequence failures (self-harm advice, a double-charge, an unsafe instruction) are in practice deliberately rare. Rarity is what makes them invisible to aggregation; gravity is what makes that invisibility intolerable.

Nor is the masking recoverable after the fact. Given only an observed conjunction rate $\widehat{p}$, the per-criterion rates are not identified: any allocation of $1 - \widehat{p}$ among the criteria that respects the union bound is consistent with the observation. Recovering them

requires per-criterion attribution at trial-record time — the wide trial record the per-criterion model demands. An archive that did not preserve per-criterion outcomes per trial cannot be decomposed retrospectively.

Per-criterion partitioning is therefore the only faithful representation of a contract whose postconditions defend against failure modes of differing consequence. Where the postconditions are genuinely interchangeable — equivalent consequence, comparable frequency, the same inputs — a single aggregated stream remains adequate, and that is exactly the $m = 1$ case recovered unchanged. A contract may still report an aggregate $\hat{p}$ as a descriptive dashboard statistic, but it is not threshold-bearing: thresholds are not derived from it, and verdicts are not framed against it.

4.5 Criterion Design Rule

The criterion is the unit of statistical inference, and a criterion may host more than one postcondition, conjoining their per-trial verdicts into a single pass/fail stream. That conjunction can reintroduce the very masking the per-criterion partition exists to prevent, one level lower: a rare, high-consequence postcondition folded in beside a frequent, low-consequence one is again absorbed into the latter's failure rate. The partition is therefore disciplined by a design rule.

> Postconditions may be hosted in a single criterion only when their failures share the same consequence class, the same remediation path, and broadly comparable expected frequency. Postconditions that differ along any of these axes belong in separate criteria.

Even where postconditions are legitimately conjoined, the per-postcondition outcomes are retained in the wide trial record demanded above, so that a criterion-level failure can be attributed to the postcondition that caused it: the conjunction governs the verdict, not what is recorded.

4.6 Clause Forms: Rate-Bounded and Categorical

Two questions about a clause must be kept apart: what kind of proposition does it state? — its **form** — and, if it carries a threshold, where did that threshold come from? — its **origin**. The form is the subject of this section; the origin axis is developed under threshold derivation. The two axes are independent, and the word empirical names a value on the origin axis — it is not the name of a form.

A **rate-bounded clause** states a proposition of the shape criterion c shall hold at rate at least p_c^*, with confidence $1 - \alpha_c$. It is discharged statistically, by the machinery this paper develops, and it is the form every criterion discussed so far has taken. Its threshold p_c^* has one of two **origins**, and the two are genuinely different inferential acts:

- **normative** (equivalently stipulated): the threshold is given — by a contract, SLA, SLO, policy, or regulation — and the question is whether the service meets it. Nothing is estimated about where the threshold should sit; it is mandated.
- **empirical**: the threshold is derived from a baseline experiment, and the question is whether the service has regressed from the rate it previously achieved.

The word empirical is reserved here for this second origin — a threshold established by experiment — and is deliberately **not** the name of the rate-bounded form, precisely because a rate-bounded clause may equally carry a normative threshold that was never measured. The frameworks keep the two as separate authoring constructs rather than two settings of one knob: a normative target is declared as meeting a required rate, and an empirical baseline as an empirical derivation. The compliance and regression paradigms of the threshold chapter are exactly these two origins seen from the decision side.

A **categorical clause** states an obligation that admits no rate threshold at all: the service shall not emit self-harm advice, shall not leak personal data, shall not produce illegal content. It is satisfied or violated in kind, not in rate. No sample budget redenominates it as a rate-bounded claim — observing zero failures in n trials yields at best a rule-of-three upper bound of $\approx 3/n$ on the failure rate, which reaches zero only in the limit — and letting a rate-bounded threshold approach 1 does not promote it into a categorical one; it merely tightens a bound that can never reach certainty. The frameworks recognise the categorical clause in exactly one form: a **zero-failures** declaration, by which the developer states the expectation while explicitly abandoning any pretence of establishing a statistical lower bound. Such a clause is evaluated observationally — it reports whether any failure was seen, never an estimated rate (see the next section).

A categorical clause is **discharged architecturally**. A dedicated component — a guardrail, a deterministic filter, a refusal classifier — is interposed between the stochastic service and its consumer, positioned to catch the prohibited outcome before it reaches anyone. That component is developed independently and carries its own distributional contract: it is tested probabilistically against an adversarial sampling — a deliberately poisoned collection of inputs constructed to provoke the prohibited outcome — and tuned until it catches them at a very high rate, established at high confidence. Once the component meets its own contract it is added to the architecture, and the developer can rest assured that even a rare prohibited outcome will not get past it. The guardrail is therefore not a criterion of the service's contract; it is a separate component, separately contracted, and its presence is what discharges the categorical obligation. The methodology, in one line:

> Tolerable failures are bounded statistically; intolerable failures are bounded architecturally; and the architecture itself is bounded statistically.

The full treatment of architectural discharge — how the commitment is declared, how the adversarial sampling's coverage is recorded, how the component's false-positive rate enters alongside its false-negative behaviour — is deferred to a dedicated chapter of the Statistical Companion. This paper introduces the form/origin distinction and the rate-bounded/categorical split, which together determine how a given failure mode is discharged.

4.7 Inferential and Observational Criteria

Two levels of outcome must be kept distinct here. On a **single sample**, a criterion's evaluation has only two possible outcomes — **PASS** or **FAIL** (a FAIL carrying the

diagnostic condition or transform/no-value reason of The Operational Outcome of a Trial above); there is no third per-sample outcome. It is only the criterion's **run-level verdict**, aggregated over the whole sampling, that admits a third value, **INCONCLUSIVE** — returned when the run cannot support a determination at all, not when some sample was hard to evaluate. A criterion reaches its run verdict in one of two modes.

An **inferential criterion** estimates the population rate p_c from the run's per-sample PASS/FAIL outcomes and tests it against a threshold. Its run verdict is PASS when the evidence clears the threshold at confidence $1-\alpha_c$, FAIL when it does not, and INCONCLUSIVE when the run cannot support a determination at all — either because the sample was too small to clear the feasibility gate at the stated threshold and confidence (the sample size n_c below the feasible minimum, or $n_c = 0$), or because the threshold is empirical in origin and no baseline rate could be found for the covariates in force at evaluation time, leaving nothing to test against. INCONCLUSIVE is therefore a statement about the run, never about any one sample. This is the mode for the question what is the true rate of behaviour c, and does it clear the demanded threshold?

An **observational criterion** asks only whether any failure was seen in the run. It estimates no parameter and carries no threshold:

$$\text{verdict}(c) = \begin{cases} \text{PASS} & \text{if } K_c = n_c \text{ and } n_c > 0 \\ \text{FAIL} & \text{if } K_c < n_c \\ \text{INCONCLUSIVE} & \text{if } n_c = 0. \end{cases}$$

A passing observational verdict at $n_c = 1000$ says exactly: no failure of c was observed in 1000 trials. It entails no bound on the true population rate. This is the honest expression of a zero-failures expectation: an inferential test against a threshold of 1.0 cannot pass at any finite sample size, since the Wilson lower bound at $\widehat{p}_c = 1$ is strictly below 1 for every finite n_c. The framework therefore does not accept 1.0 as an inferential threshold and does not silently convert an observational criterion into one. For transparent reporting only, a passing observational verdict may be annotated with the rule-of-three upper bound of $\approx 3/n_c$ under an explicitly stated i.i.d. Bernoulli model, and high-consequence criteria may report NO FAILURE OBSERVED in place of PASS — a label change that leaves the verdict semantics untouched.

The INCONCLUSIVE verdict must not be confused with a per-trial transform/no-value FAIL: a non-evaluable response is a FAIL of the criterion with a recorded reason, whereas INCONCLUSIVE is a non-determination at the verdict level — the procedure could render neither PASS nor FAIL. The two live at different levels and carry different meanings (Statistical Companion §1.4.5, §1.4.5a).

Observational criteria are the methodology's vehicle for the zero-failures evidence that supports an architectural commitment: a guardrail's false-negative behaviour over an adversarial sampling is reported as an observational verdict, read alongside the commitment it supports rather than as a discharge of the categorical clause in its own right.

5. Deriving Thresholds from Baselines

As the discussion of clause forms noted, a rate-bounded criterion's threshold p_c^* has one of two origins, and the same two origins were the ”two operational modes” of the single-predicate model. In the first, the threshold is stipulated externally — by a contract, service level agreement, regulatory requirement, or organisational policy. Such a threshold is a normative claim: it expresses what the service ought to achieve, independently of what it has been observed to achieve. No experiment is required; the statistical question is simply whether the service meets the mandated requirement.

In the second mode, no external threshold exists. The service's acceptable performance level is not known in advance and must be discovered empirically. This is the common case for systems whose behaviour is inherently stochastic, where the developer cannot stipulate a success rate from first principles but must instead observe what the service actually achieves under controlled conditions. Services built on Large Language Models are natural candidates for this mode: an LLM invoked with a fixed system prompt, model version, and temperature setting may be reasonably expected to exhibit a consistent level of stochasticity over time, making the baseline estimate a stable foundation for subsequent evaluation — provided the operational profile remains comparable.

5.1 The Baseline Experiment

To establish an empirical threshold, a baseline experiment is conducted: the service is executed n_{exp} times under controlled conditions, and the observed success rate $\hat{p}_{\text{exp}} = k/n_{\text{exp}}$ is recorded. This point estimate serves as the best available characterisation of the service's current behaviour. It is important to note, as Musa [9] established in the context of software reliability engineering, that any such empirical estimate is meaningful only relative to the distribution of inputs under which the service is exercised. A baseline measured under one operational profile may not generalise to another — an insight that motivates the covariate tracking discussed in later sections.

Because every criterion is exercised against the same shared sampling, a single baseline experiment records a success rate per criterion: the baseline is vectorised across the contract's criteria, $\{\hat{p}_{\text{exp},c}\}$, and a threshold p_c^* is derived for each criterion independently by the construction below. A criterion that requires evidence about a different input distribution is measured in its own experiment, against its own sampling.

However, $\hat{p}_{\text{exp}}$ cannot be used directly as a threshold for subsequent evaluations. The point estimate is subject to sampling variability: a different run of the same experiment would yield a different value of $\hat{p}$. Setting $p_{\text{min}} = \hat{p}_{\text{exp}}$ would amount to requiring the service to meet a standard that is an artefact of a particular sample, not a stable property of the service. In practice, this leads to frequent false alarms — tests that reject a service whose true success probability has not changed, simply because the test sample happened to fall below the experimental estimate.

5.2 Conservative Threshold Derivation

The solution is to derive the threshold not from the point estimate itself, but from a conservative lower bound on the success probability the baseline supports at the sample size the test will actually use. The threshold for a criterion is the one-sided Wilson lower bound computed from the baseline estimate $\hat{p}_{\text{exp}}$, the **test** sample size n_{test}, and the confidence level α:

$$p_c^* = L(\hat{p}_{\text{exp}}, n_{\text{test}}, \alpha)$$

The test then compares its own observed rate $\hat{p}_{\text{test}}$ against this threshold, and the criterion passes when $\hat{p}_{\text{test}}$ clears it. The test sample size — not the experiment's — enters the bound: the threshold answers given a test of size n_{test}, how far below the baseline could the observed rate fall by sampling variation alone, at confidence $1-\alpha$?

This yields a threshold with the following property: if the service's true success probability is at least the baseline estimate, a test of size n_{test} rejects falsely with probability at most α. The threshold sits below $\hat{p}_{\text{exp}}$, absorbing the sampling variability a test of that size will exhibit, so that only genuine degradation — not sampling noise — triggers a failure.

The gap between $\hat{p}_{\text{exp}}$ and the threshold p_c^* depends on the **test** sample size. A larger test produces a tighter bound and hence a threshold closer to the baseline rate; a smaller test produces a wider bound and a correspondingly more conservative — lower — threshold. This is the correct behaviour: the smaller test is the noisier estimate and warrants the more forgiving bar, so that its sampling variation alone does not trip a false alarm.

This paper states the construction; the Statistical Companion treats it in depth. The Wilson score lower bound is derived and its exclusive use justified in companion §2.3.1 (Wilson Score Interval, including Why Wilson Exclusively?), and its application to threshold derivation — the one-sided lower bound $L(\hat{p}_{\text{exp}}, n_{\text{test}}, \alpha)$ together with the integer-cutoff decision form it induces — is set out in companion §3.4 (Threshold Calculation), with reference values in §3.5.

5.3 Choice of Confidence Bound

The lower bound $L(\hat{p}, n, \alpha)$ used throughout this framework is the Wilson score interval [10]. This choice warrants brief justification, since alternative methods exist and differ in important respects.

The Wilson score interval is constructed by inverting the score test for a binomial proportion. For n observations with observed success rate $\hat{p}$ and critical value $z = z_\alpha$, the one-sided lower bound is:

$$L(\hat{p}, n, \alpha) = \frac{\hat{p} + \frac{z^2}{2n} - z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

This bound has several properties that make it well-suited to the present framework: it produces valid bounds in $[0, 1]$ for all values of $\hat{p}$ and all sample sizes, it is far better calibrated than the Wald interval in the small-sample and near-boundary regimes that matter for high-reliability claims — though, unlike the exact Clopper–Pearson interval, it does not guarantee exact coverage at every p and n — and it remains well-defined at the boundary cases $\hat{p} = 0$ and $\hat{p} = 1$.

Its value is that it stays well-behaved even in the pathological instances where simpler methods fail. The Wald normal approximation, the most commonly seen alternative, collapses to a zero-width interval when an experiment delivers a 100% success rate — precisely the regime a well-performing service occupies — and can place bounds outside $[0, 1]$ near the extremes. The Wilson bound has no such failure modes. The full comparison with the Wald, Clopper–Pearson [11], and Bayesian alternatives, and the rationale for using Wilson exclusively, is given in the Statistical Companion §2.3.1 (Why Wilson Exclusively?; see also Brown et al. [12] and Agresti and Coull [13]); it is not reproduced here.

5.4 Stipulated and Derived Thresholds in a Common Framework

Stipulated and empirically derived thresholds share the same Wilson machinery but apply it in mirror-image ways. A **stipulated** (normative) threshold p_{req} is given; the test computes the Wilson lower bound on its own sample, $L(\hat{p}_{\text{test}}, n_{\text{test}}, \alpha)$, and passes when that bound clears p_{req} — affirmative evidence that the service meets the requirement. An **empirically derived** threshold is itself a Wilson lower bound, $p_c^* = L(\hat{p}_{\text{exp}}, n_{\text{test}}, \alpha)$, computed from the baseline at the test sample size; the test passes when its observed rate $\hat{p}_{\text{test}}$ clears that threshold — evidence of no degradation from the measured baseline. The two procedures control different errors and carry different meanings (companion §3.2): a stipulated failure is evidence the service does not meet an external requirement, while a derived failure is evidence it has degraded from its measured baseline.

6. Temporal Stochasticity

The preceding sections developed the distributional contract for functional stochasticity, in which each execution yields a binary outcome and the contractual assertion is over the success probability. Temporal stochasticity requires a fundamentally different treatment. Each execution yields a continuous quantity — its duration — and the contractual assertion is not over a success rate but over the shape of a distribution.

6.1 Why Latency Demands Distributional Reasoning

A single observed response time tells us almost nothing about a service's temporal behaviour. Service latency distributions are typically right-skewed, often multimodal, and frequently heavy-tailed. A service with a 200ms mean may have a 95th percentile of 350ms or of 2000ms — the mean alone cannot distinguish the two. What matters operationally is not the average case but the tail: the experience of the worst-served fraction of requests. This is why latency contracts are expressed not in terms of means but in terms of percentiles.

6.2 The Latency Population

Not all executions contribute to the latency distribution. Executions that failed the contract's correctness criteria — whether because a postcondition did not hold or because no testable value was produced (a transform/no-value failure) — are excluded from the latency population. A fast failure (an immediate validation rejection at $t \approx 0$) and a slow failure (a timeout at $t = 30{,}000$ms) both reflect error paths, not the latency of successful operation. Including them would contaminate the distribution with durations that are artefacts of failure modes rather than measurements of service responsiveness. (A true catastrophe that aborts the run, as distinct from these in-band failures, produces no sample at all.)

The latency distribution is therefore conditional on success. Let $\mathcal{L} = \{t_i : X_i = 1\}$ be the set of durations from executions that satisfied the contract's correctness criteria, and let $n_s = |\mathcal{L}|$. The distributional contract over temporal stochasticity applies to this conditional distribution $T \mid X = 1$, not to the unconditional distribution over all attempts. Functional stochasticity and temporal stochasticity thus describe two complementary aspects of the same set of executions: the probability of success, and the latency distribution given success.

Conditioning on success is the correct modelling choice, but it creates a hazard that must be named explicitly. Because the latency contract applies only to the successful subset, a service could in principle improve its observed percentiles by converting slow-successes into fast-failures — moving mass from the right tail of $T \mid X = 1$ into the failure column entirely. This is not a defect of the conditioning; it is the reason the correctness criteria and the latency contract are evaluated **jointly, with logical conjunction**. A service cannot trade correctness for latency under the composite verdict, because each correctness criterion remains independently enforced at its own threshold. An operator writing latency contracts should ensure the accompanying correctness threshold is tight enough that this trade cannot be silently exploited; a latency improvement accompanied by a correctness regression is not an improvement of the overall contract.

Organisations that require an unconditional response-time guarantee — that is, a bound on latency marginalised over all attempts, including those that fail — should express this as a joint constraint on the correctness criteria and the latency contract, rather than attempt to redefine the latency population itself. The conditional distribution $T \mid X = 1$ is the only population for which non-parametric percentile inference over successful-response latency is meaningful.

6.3 Empirical Percentile Estimation

Because latency distributions resist parametric characterisation, the contract is assessed using non-parametric empirical percentiles computed directly from the observed order statistics. No assumptions are made about the shape of the underlying distribution.

Let $t_{(1)} \leq t_{(2)} \leq \cdots \leq t_{(n_s)}$ be the order statistics of the successful latencies. For a percentile level $p \in (0, 1]$, the empirical percentile is:

$$Q(p) = t_{(\lceil p \cdot n_s \rceil)}$$

This is the nearest-rank method: the p-th percentile is the smallest observed value such that at least a proportion p of observations fall at or below it.

Strictly speaking, the order-statistic arguments developed below for threshold derivation assume a continuous latency distribution, under which ties occur with probability zero. Wall-clock latencies reported at millisecond resolution induce ties, and the constructions remain valid — the bounds become marginally more conservative under ties, not less — at the cost of a small loss of tightness. Practitioners who require tight bounds should report latencies at microsecond resolution.

6.4 The Latency Assertion

A latency contract specifies a set of percentile constraints:

$$\mathcal{C} = \{(p_j, \tau_j) : j = 1, \ldots, m\}$$

where p_j is a percentile level (e.g., 0.50, 0.90, 0.95, 0.99) and τ_j is the maximum acceptable value at that percentile. For each constraint, the assertion evaluates:

$$\text{PASS}_j \iff Q(p_j) \leq \tau_j$$

The overall latency contract is satisfied if and only if all constraints are met:

$$\text{PASS}_{\text{latency}} = \bigwedge_{j=1}^{m} \text{PASS}_j$$

As with functional stochasticity, the threshold τ_j may be stipulated externally — by an SLA specifying, say, that the 95th percentile latency must not exceed 500ms — or derived empirically from a baseline experiment. The same dichotomy of mandated versus empirical thresholds applies, with the same interpretive distinction: a breach of a stipulated threshold is evidence of non-compliance; a breach of an empirically derived threshold is evidence of degradation.

6.5 Threshold Derivation for Latency

When τ_j is derived from a baseline, the observed percentile $Q_{\text{baseline}}(p_j)$ is, like $\hat{p}_{\text{exp}}$ in the functional case, a point estimate subject to sampling variability. Using it directly as a threshold would again invite false alarms. The task is to construct a conservative upper confidence bound on the baseline percentile — the non-parametric analogue of the Wilson-score lower bound used in the functional case. Statistical symmetry between the two sides of the distributional contract is desirable, and it is achievable: the construction below is exact, distribution-free, and integer-valued by construction.

For i.i.d. successful-response latencies drawn from a continuous distribution, the rank of the population quantile $Q(p_j)$ within the sorted sample is distributed as a binomial random variable — specifically, the number of observations at or below $Q(p_j)$ is $\text{Bin}(n_s, p_j)$. This classical result (see Conover [14] or Hollander and Wolfe [15]) yields an exact one-sided upper confidence bound on the true quantile:

$$\tau_j = t_{(k_j)} \qquad \text{where} \qquad k_j = \min\{k : \mathbb{P}(\text{Bin}(n_s, p_j) \geq k) \leq \alpha\}$$

Equivalently, k_j is the smallest rank at which the probability of observing k or more sample points at or below the true $Q(p_j)$ falls to at most α. In practice this is the `qbinom(1 - α, n_s, p_j) + 1` order statistic, clamped below at the point-estimate rank $\lceil p_j \cdot n_s \rceil$ and above at n_s. The threshold is therefore the k_j-th smallest observed latency in the baseline — an observed value in milliseconds, requiring no rounding, no normal approximation, and no density estimate.

The construction has several properties worth naming. Because it is an observed order statistic of an integer-valued baseline, the threshold is integer-valued without any ceiling operation. It is monotone in both α and p_j: higher confidence and higher percentiles yield higher ranks, and therefore looser (more conservative) thresholds. It is never tighter than the raw baseline observation: $k_j \geq \lceil p_j \cdot n_s \rceil$ holds by construction. And it degenerates gracefully at small sample sizes: when n_s is too small for the confidence level to be resolved, k_j saturates at n_s and $\tau_j = t_{(n_s)} = \max$, which signals infeasibility to the feasibility gate rather than producing a silently misleading threshold.

One caveat is important and deserves to be stated plainly. The construction is a confidence bound on the true baseline quantile $Q_{\text{true}}(p_j)$ — it is not a prediction interval for the next experiment's observed $\hat{Q}_{\text{test}}(p_j)$. These are different objects. When the test-experiment sample size is substantially smaller than the baseline, the test-side sampling variance contributes additional uncertainty that the confidence bound alone does not absorb, and false-positive rates will exceed the nominal α by a factor that grows as the sample-size imbalance grows.

The construction is nonetheless well-posed: a breach of τ_j remains statistically meaningful — it exceeds a legitimate upper bound on the baseline — but operators who require tight false-positive calibration across baseline and test experiments should size the test experiment comparably to the baseline. An alternative construction yielding a true predictive interval for future observed percentiles would require either a parametric assumption about the latency distribution or a full non-parametric tolerance-interval treatment in the sense of Wilks [16]; the mavai methodology chooses the simpler confidence-bound construction and documents its limitation explicitly.

The earlier draft of this paper used the standard error of the mean as a proxy for percentile uncertainty, $\hat{\tau}_j = Q_{\text{baseline}}(p_j) + z_\alpha \cdot s/\sqrt{n_s}$. That construction was an engineering approximation, not a formal non-parametric bound, and understated tail-percentile uncertainty for heavy-tailed distributions by a factor that grew with skewness — precisely the distributions for which percentile-based latency contracts are most often written. It has been superseded by the binomial order-statistic bound

above. The older formula is noted here for continuity with prior versions of this document and is not recommended for use.

A further consequence of the empirical approach is that small sample sizes trivially exclude higher percentiles from meaningful evaluation. The p-th percentile requires at least $\lceil 1/(1-p) \rceil$ observations before it is distinct from the sample maximum: for $p = 0.95$ this is 20, for $p = 0.99$ it is 100. When n_s falls below such a threshold, the corresponding percentile constraint cannot be assessed at all. And even at the boundary — $n_s = 100$ for $p = 0.99$, for instance — the percentile estimate rests on a single observation separating it from the maximum, which is hardly a basis for a meaningful contractual judgement. In practice, reliable percentile estimation requires n_s to be substantially larger than the theoretical minimum, so that the estimate is determined by a region of the distribution rather than by an isolated order statistic.

6.6 The Composite Verdict

With both dimensions defined, the contract's verdict combines its per-criterion functional verdicts with its latency verdict. Crucially, this is not a single flattened boolean: a flat PASS/FAIL surface would obscure exactly the per-criterion outcomes the criterion decomposition exists to expose. The contract verdict is a **structured tuple** over the per-criterion verdicts $V_c \in \{\text{PASS}, \text{FAIL}, \text{INCONCLUSIVE}\}$, with the structural composite

$$V_{\text{contract}} = \begin{cases} \text{PASS} & \text{if } \forall\, c : V_c = \text{PASS} \\ \text{FAIL} & \text{if } \exists\, c : V_c = \text{FAIL} \\ \text{INCONCLUSIVE} & \text{otherwise,} \end{cases}$$

and the per-criterion verdicts retained in full on the composite so that a consumer reads the conclusion and its supporting evidence in one place. The latency contract enters as one further criterion-like term in the conjunction: a service is acceptable only if every functional criterion passes and the latency constraints hold.

A service must therefore be both correct and responsive. A payment API that succeeds 99.5% of the time but takes 30 seconds for 1% of requests fails its contract just as surely as one that returns incorrect results — and, under the decomposition, a contract in which a rare safety criterion has regressed fails even when an aggregate rate would still look healthy. The dimensions and criteria are assessed independently, each by its own statistical framework, and all must pass for the service to be judged acceptable.

Because a contract aggregates many per-criterion tests, its false-alarm budget is itself an aggregate. The methodology discloses it as a **Type-I envelope**: by the union bound, the probability of some inferential criterion raising a false alarm is at most the sum of the per-criterion α_c over the relevant family of criteria, and this bound holds under arbitrary dependence among the criteria.

The envelope is reported on the composite verdict as a disclosed property, not driven to a target by shrinking the per-criterion α_c: a safety criterion held at $\alpha = 0.001$ keeps that level because the consequence of falsely accepting a safety regression

demands it, and uniformly dividing the α_c would sap exactly the power such a criterion most needs. The Statistical Companion (§1.4.6) gives the envelope its precise form, separating the false-compliance and false-degradation-signal directions, which control different error events and may be set at different magnitudes.

Availability is no longer a separate orthogonal sub-contract here: under the accounting above it is reflected end-to-end in every quality criterion, and a contract that wants it visible declares it as its own criterion, which then takes its place among the per-criterion verdicts of the tuple.

7. Sample Size and Feasibility

Not every distributional contract can be meaningfully assessed with a given number of samples. The relationship between sample size, threshold, and confidence is not a free choice — it is a system of constraints in which fixing any two determines the third. This concern echoes a theme in the software reliability literature: Frankl, Hamlet, Littlewood, and Strigini [17] argue that testing methods should be evaluated in terms of the reliability they ultimately deliver in operation, not merely in terms of fault counts, and Hamlet and Taylor [18] demonstrate that success on a set of carefully chosen examples does not, by itself, justify confidence in future behaviour under realistic conditions. These results become sharper still for stochastic services, where the property of interest is inherently probabilistic and no single execution can be decisive. This section examines the three ways in which the constraint system can be resolved.

7.1 The Fundamental Constraint

A distributional contract involves three quantities that are bound together by the statistical framework: the sample size n, the threshold $p_{\min}$, and the confidence level $1 - \alpha$. It is not possible to simultaneously achieve an arbitrarily tight threshold, high confidence, and a small sample. This is not a limitation of any particular method; it is a fundamental property of statistical inference. Any operational approach to parameterising a distributional contract must fix two of these quantities and accept that the third is determined.

Under the criterion decomposition this constraint is per criterion: each criterion c binds its own triple (n_c, p_c^*, α_c) and has its own feasibility gate. The criteria of one experiment share a single sampling, so they share its size — each criterion's trial count n_c equals the sampling's size N, since every sample is seen by every criterion. The **limiting criterion** is the one whose triple demands the largest N, and it sets the experiment's required sample size; a low-baseline or tightly-thresholded criterion can therefore dominate the cost of the whole experiment, which per-criterion feasibility gates surface honestly where an aggregate would hide it. A criterion that needs evidence about a different input distribution does not inflate this N; it runs in a separate experiment with its own sampling, and a contract with E experiments has total sample budget bounded by $\sum_e N_e$.

Three natural approaches emerge, each corresponding to a different practical starting point.

7.2 Sample-Size-First: Cost-Driven

In many operational settings, the sample size is fixed by external constraints: the cost of each execution, the time available in a continuous integration pipeline, or the rate limits imposed by an external service. The question becomes: given a fixed n and a threshold p_{min}, what confidence does the resulting evaluation achieve?

The confidence is determined implicitly by the Wilson lower bound. For a given n and p_{min}, the achievable confidence is the level $1-\alpha$ at which the lower bound $L(1, n, \alpha) = p_{\text{min}}$ — that is, the confidence at which even a perfect observation ($\hat{p} = 1$) would just barely satisfy the contract. If the achievable confidence is unacceptably low, the evaluation remains valid but its evidential weight is limited. It may detect gross violations but will lack the sensitivity to distinguish small degradations from sampling noise.

7.3 Confidence-First: Risk-Driven

When the consequence of a false verdict is severe — in safety-critical systems, compliance audits, or contractual obligations — the confidence level and detection capability are the primary requirements. The question becomes: given a threshold p_{min}, a desired confidence $1 - \alpha$, and a minimum detectable effect δ, how many samples are required?

The minimum detectable effect is essential: it specifies the smallest degradation worth detecting, expressed as a drop from p_{min} to $p_{\text{min}} - \delta$. Without it, the sample size has no finite answer — detecting an arbitrarily small degradation requires an arbitrarily large sample.

Given these parameters, the required sample size for a one-sided test with power $1 - \beta$ is:

$$n = \left(\frac{z_\alpha \sqrt{p_{\text{min}}(1 - p_{\text{min}})} + z_\beta \sqrt{(p_{\text{min}} - \delta)(1 - (p_{\text{min}} - \delta))}}{\delta} \right)^2$$

where z_α and z_β are the critical values corresponding to the desired significance and power levels respectively.

The practical consequence is that stringent contracts demand large samples. Verifying a 99.9% threshold with the ability to detect a 0.1% degradation at 95% confidence and 80% power requires on the order of 8,000 samples. This is not excessive; it is the honest cost of making a statistically defensible claim about a service operating at that level of reliability.

7.4 Threshold-First: Baseline-Anchored

In the third approach, the threshold is taken directly from the empirical baseline — $p_{\text{min}} = \hat{p}_{\text{exp}}$ — without the conservative adjustment described in the section on threshold derivation. Its behaviour depends critically on how subsequent runs are compared against it, and two cases must not be conflated.

If runs are compared by raw point estimate — $\hat{p}_{\text{test}} \geq \hat{p}_{\text{exp}}$ — then a service whose true rate has not changed falls below the baseline estimate with probability on the order of one half, by sampling variation alone (subject to discreteness and unequal sample sizes).

If instead the inferential rule used throughout this paper is applied — requiring the lower confidence bound $L(\hat{p}_{\text{test}}, n_{\text{test}}, \alpha) \geq p_{\text{min}}$ — the behaviour is far stricter, not a coin toss: at a true rate equal to $p_{\text{min}} = \hat{p}_{\text{exp}}$, the lower bound lies below the threshold in the great majority of runs, so the contract fails almost always. Either way, a raw point estimate is the wrong quantity to enforce against.

This approach is retained only as a descriptive or exploratory mode — useful when the operator is deliberately probing the trade-offs between threshold, sample size, and false-positive rate, and is willing to accept the consequences. It should not be used where the verdict carries operational weight. An enforceable empirical threshold should be a conservative baseline-derived bound (as in Conservative Threshold Derivation above), or regression should be assessed by an explicit two-sample non-inferiority procedure rather than by anchoring on the point estimate.

7.5 The Feasibility Gate

Regardless of which approach is taken, there exists for any given threshold and confidence level a minimum sample size below which no observation — not even a perfect one — can satisfy the contract. If every execution succeeds ($\hat{p} = 1$), the Wilson lower bound is:

$$L(1, n, \alpha) = \frac{n}{n + z^2}$$

For this bound to reach p_{min}, the sample size must satisfy:

$$n \geq N_{\text{min}} = \left\lceil \frac{p_{\text{min}} \cdot z^2}{1 - p_{\text{min}}} \right\rceil$$

If $n < N_{\text{min}}$, the contract is infeasible: the evaluation cannot produce a positive verdict under any outcome. This can and should be detected before any samples are collected. Proceeding with an infeasible configuration wastes resources on an evaluation whose conclusion is predetermined.

The feasibility gate scales steeply with the threshold. At the default confidence level of 95% ($z \approx 1.645$):

p_{min}	N_{min}
0.80	11
0.90	25
0.95	52
0.99	268
0.999	2,704

$p_{\min}$	$N_{\min}$
0.9999	27,058

A contract requiring $p \geq 0.9999$ cannot be verified with fewer than 27,058 samples. This is not a deficiency of the method; it is the inescapable cost of making a credible assertion about a service operating at four nines of reliability.

8. The mavai Project Family: Implementations of the Distributional Contract

The preceding sections developed the distributional contract as a statistical model. This section describes the mavai project family, which implements the model as practical tooling for developers working with stochastic systems.

The mavai project was created in response to a conspicuous gap in the testing landscape. The shortcomings of existing approaches to testing stochastic systems have been recognised in the literature. Christensen et al. [19] observe that ”in practice, testers resort to running probabilistic programs an arbitrary number of times; hoping to trigger existing bugs, but without any guarantees,” and propose ProbTest, a black-box unit testing method that uses the coupon collector's problem to determine sample sizes for bug detection in probabilistic programs. Their work confirms the need for statistically principled approaches and demonstrates that the problem admits tractable solutions, though their focus is on bug-finding rather than on contractual quality assurance over deployed services.

The advent of Large Language Models brought stochastic behaviour from the margins of software engineering — where it had been treated as accidental and undesirable — to the centre. Services built on LLMs are inherently non-deterministic: the same input genuinely produces different outputs, and the distribution of those outputs is the service's behaviour. Yet the available testing tools offered no principled way to express, measure, or enforce quality expectations over such services. The mavai project family addresses this gap by providing direct operational realisations of the distributional contract across multiple language ecosystems.

The methodology is implemented as language-native frameworks, each idiomatic to its ecosystem:

Framework	Language	Integration	Repository
punit	Java	JUnit 5 extension	The reference implementation
feotest	Rust	Cargo test integration	Idiomatic Rust, not a port
baseltest	Python	pytest plugin (planned)	—

Each framework provides a direct operational realisation of the distributional contract as described in this paper. Their features correspond to the elements of the

model.

8.1 Expressing a Distributional Contract

Each framework gives the developer the means to express a distributional contract as a declarative specification. The functional dimension is defined through one or more **criteria**, each hosting one or more postconditions — syntactic validity, presence of required fields, schema adherence, semantic correctness, absence of disallowed content. Postconditions whose failures carry materially different consequences are placed in separate criteria, each its own statistical stream with its own threshold and confidence level, rather than fused into a single composite assertion; a single criterion is the right expression only where the postconditions are genuinely interchangeable. The developer may additionally declare latency thresholds — percentile-level bounds to which the service must adhere — expressing the temporal dimension of the contract.

All frameworks evaluate both dimensions using the statistical methods described in this paper: a per-criterion Bernoulli model with Wilson score lower bounds for functional stochasticity, and empirical percentile analysis for temporal stochasticity. Each criterion yields its own verdict, and the frameworks report the structured composite over those verdicts — never a single flattened boolean — together with the disclosed Type-I envelope; the contract passes only if every criterion and the latency constraints pass.

The move to a distributional contract also reframes what it means for a postcondition to fail. In Meyer's original formulation, a violated postcondition is evidence of a defect in the supplier: the routine failed to honour its obligation, and the appropriate response is to locate and fix the bug. Under a distributional contract this interpretation is no longer available at the level of an individual execution. A single execution on which a postcondition is not satisfied does not, by itself, establish a defect — it is indistinguishable from a sample outcome drawn from the variability the contract explicitly admits.

The notion of contract failure is therefore lifted along with the postcondition: a failure is no longer the falsification of a postcondition on one execution, but the failure of a criterion, over a principled number of executions, to deliver the desired outcome with sufficient probability. Accordingly, the frameworks treat an individual failing execution as data, not as an alarm, and reserve the contract-failure verdict for the statistical assessment of the population as a whole.

8.2 Stipulated and Empirical Thresholds

Both threshold sources are supported across all implementations. A developer may declare a stipulated threshold directly — from an SLA, regulatory requirement, or organisational policy — or may conduct a baseline measurement experiment from which the framework derives a conservative threshold using the Wilson lower bound. The baseline result is recorded as a persistent specification artefact, capturing the observed success rate, sample size, and derived threshold together with the conditions under which the baseline was established.

8.3 The Feasibility Gate

All implementations enforce the feasibility constraint described in this paper. Before any samples are executed, the framework determines whether the configured sample size can in principle support a positive verdict at the required confidence level. If it cannot, the evaluation is rejected as infeasible — a configuration error, distinct from a system failure. This prevents the waste of resources on evaluations whose conclusion is predetermined.

8.4 Early Termination

In some cases, the final verdict can be determined before all planned samples have been executed. If enough failures have been observed that the lower bound cannot reach p_{min} even if all remaining samples pass, continued execution is pointless. Conversely, if enough successes have been observed that the lower bound will exceed p_{min} regardless of remaining outcomes, the verdict is already determined. The frameworks detect both conditions and terminate early.

Each additional execution beyond the point at which the verdict is fixed consumes resources with real-world cost: LLM token spend billed per call, CPU and memory on the host running the test, network bandwidth to the service under test, energy and the carbon footprint it implies, and — in rate-limited environments — quota that may be needed elsewhere. These costs scale linearly with the number of samples and can be substantial: a single probabilistic test of a large language model may consume tens of thousands of tokens, and a test suite running in continuous integration may execute thousands of such tests per day.

Continuing to sample once the verdict is mathematically determined is therefore an active waste of non-renewable budget. The frameworks treat early termination as a first-class obligation rather than an optional optimisation, on the pragmatic grounds that a methodology which ignores the operational cost of its own evaluations is unlikely to be adopted, and less likely still to be sustained.

8.5 Operational Safeguards

The Bernoulli model assumes independence between executions and stationarity of the success probability across the sampling window. These assumptions are not given by the operational environment; they must be actively maintained. The frameworks provide several mechanisms toward this end. A warmup facility allows initial observations to be discarded, mitigating cold-start non-stationarity caused by caches, connection pools, or runtime compilation.

A particular threat to stationarity arises from exogenous covariates: factors external to the service under test that influence the success probability without being part of the input. In any population sampling exercise — and Bernoulli trial sampling is no exception — the validity of comparisons between samples depends on the conditions under which those samples were drawn. If the baseline was established under one set of conditions and the evaluation is conducted under another, the two samples are drawn from different populations, and the comparison is statistically meaningless regardless of the sample sizes involved.

Deployment region, time of day, model version, infrastructure configuration, and upstream service behaviour are all examples of exogenous covariates that can shift the success probability between a baseline and a subsequent evaluation. The frameworks address this by associating covariate declarations with baselines, making contextual factors explicit and auditable, and issuing warnings when conditions at evaluation time differ from those recorded in the baseline.

Baseline expiration tracking alerts operators when a specification may no longer represent current service behaviour. These mechanisms do not guarantee that the Bernoulli assumptions hold — that is impossible in practice — but they make violations visible rather than allowing them to silently undermine inference.

The covariate problem points to a deeper reality: for many stochastic services, behaviour is environmentally dependent in ways that cannot be fully captured at baseline time. A service that meets its distributional contract in a test environment may violate it in production, not because the service itself has changed but because the operating conditions have. It is therefore insufficient, in many cases, to rely on one-time evaluations conducted in controlled settings. What is needed is a mechanism for probing the service's behaviour in situ — in the environment where it actually operates, under the conditions it actually encounters. Sentinel capabilities within the mavai family address this requirement, providing lightweight runtime agents that evaluate the distributional contract continuously against the live deployment, detecting degradation as it occurs rather than after the fact.

8.6 The Experiment-to-Test Workflow

The frameworks operationalise the measure-then-enforce workflow through a structured progression. A measurement experiment executes the service a large number of times under controlled conditions, producing a specification. Subsequent evaluations run with fewer samples and are assessed against the threshold derived from the specification. The use case — the reusable unit that invokes the service and evaluates the result — is shared between experiments and tests, ensuring that the contract's criteria are applied consistently.

8.7 Conformance Across Implementations

Because the distributional contract is a statistical model, not a language-specific construct, all mavai framework implementations must produce identical statistical results for the same inputs. The mavai-R project provides the verification mechanism: it generates canonical reference datasets using R — the gold standard for statistical computing — against which each framework validates its statistics engine. This ensures that a distributional contract evaluated by punit in Java produces the same verdict as one evaluated by feotest in Rust, within stated floating-point tolerances.

9. Conclusion

Meyer's Design by Contract gave software engineering a powerful organising principle: the obligations between caller and callee, expressed as preconditions, postconditions, and invariants, and enforced deterministically at runtime. For over three

decades it has stood as a beacon for those who care about software correctness. Still, the systems it addressed were, by and large, deterministic: a postcondition either held or it did not, and a single counterexample was definitive evidence of a defect.

Yet stochastic systems have always existed — network services subject to variable latency, randomised algorithms, probabilistic data structures, simulations — and they lie outside the deterministic scope Design by Contract was built for. A deterministic postcondition cannot express ”succeeds at least 95% of the time” or ”responds within 500ms at the 99th percentile”. These are inherently distributional claims, and no amount of Boolean logic over individual executions can capture them. Design by Contract never set out to address this — its concern was deterministic correctness — and that boundary could be overlooked as long as stochastic behaviour remained marginal. The rise of Large Language Models has made it impossible to ignore: when non-determinism is not a defect but the defining characteristic of the technology, the absence of a principled contractual framework for stochastic systems becomes an acute gap.

This gap can only be addressed by a probabilistic approach. When a service's behaviour is inherently variable, a single execution cannot characterise its quality, a single failure cannot condemn it, and a single success cannot vindicate it. The postcondition must be lifted from a Boolean predicate over an individual execution to a statistical assertion over a population of executions.

This paper has proposed such a lifting, drawing on established traditions — statistical model checking [4][5][20], black-box probabilistic verification [7][8], and software reliability engineering [9][17][18] — while recasting their insights in a contractual form suited to day-to-day software engineering practice. The distributional contract preserves the contractual structure of Design by Contract — the notion that a service has obligations to its consumers that can be formally stated and objectively assessed — while replacing pointwise evaluation with statistical inference.

The contract is expressed along two orthogonal dimensions: functional stochasticity, assessed via a Bernoulli model and the Wilson score lower bound, and temporal stochasticity, assessed via empirical percentile analysis. Within the functional dimension the contract decomposes further into separately-contractual criteria, each its own stream, because a single aggregate rate provably masks movement in the rare-but-consequential criteria a contract most needs to watch; the per-criterion verdicts compose into a structured composite carrying a disclosed Type-I envelope.

A contract's obligations also divide by form: rate-bounded clauses, bounded statistically by the machinery above, and categorical clauses, bounded architecturally and evidenced by an observational zero-failures criterion over an adversarial sampling. A rate-bounded clause's threshold is, on a separate axis, either normative in origin — stipulated by an SLA, SLO, policy, or regulation — or empirical, derived conservatively from a measured baseline; empirical names that origin, not the clause. Every criterion is subject to the same feasibility constraints that govern any honest statistical claim.

The model is deliberately conservative in its statistical choices. The Wilson bound was preferred not because it is the most sophisticated available technique, but because it is correct across the full parameter space and honest about the level of precision

that the underlying Bernoulli approximation can actually support. False precision — a more elaborate statistical treatment applied to a model whose assumptions are only approximately satisfied — was judged a greater risk than the modest loss of tightness that conservatism entails.

The mavai project family demonstrates that the distributional contract is not merely a theoretical construct but a practical tool. The frameworks translate each element of the model — the postcondition, the criterion, the threshold, the confidence bound, the feasibility gate, the baseline specification — into declarative constructs that a developer can express and a runtime can enforce, across multiple language ecosystems. Sentinel capabilities extend the model beyond one-time evaluation to continuous in-situ monitoring, acknowledging that stochastic services are environmentally dependent and that a contract verified under controlled conditions may not hold in production.

Several questions remain open. The treatment of exogenous covariates in this paper is operational rather than formal: covariates are declared and tracked, but there is no statistical framework for adjusting inference when covariate drift is detected. The latency threshold construction is a confidence bound on the baseline quantile rather than a full predictive interval for future test-experiment percentiles; the gap between the two matters when baseline and test sample sizes are materially different, and closing it would require either a distributional assumption or a full non-parametric tolerance-interval treatment.

The architectural discharge of categorical clauses is introduced here but not fully developed: how an architectural commitment is declared, how the coverage of an adversarial sampling is recorded, and how a guardrail's false-positive rate enters the contract alongside its false-negative rate are deferred to a forthcoming chapter of the Statistical Companion. And the independence assumption, while supported by operational safeguards, is ultimately unverifiable from the data alone.

These are not deficiencies to be apologised for; they are honest boundaries of what a practical framework can achieve. The distributional contract does not claim to resolve the fundamental difficulty of reasoning about stochastic systems. It claims only to provide a disciplined, statistically grounded basis for doing so — one that makes its assumptions explicit, its limitations visible, and its verdicts defensible.

10. References

[1] Meyer, B. ”Applying 'Design by Contract'.” IEEE Computer, vol. 25, no. 10, October 1992, pp. 40–51.

[2] Meyer, B. Object-Oriented Software Construction, 2nd ed. Prentice Hall, 1997.

[3] Mannion, M. F. ”Statistical Companion: Formal Statistical Foundations for the mavai Methodology.” The mavai project family, 2026. Available at https://github.com/mavai-org/mavai-R/blob/main/docs/STATISTICAL-COMPANION.md.

[4] Younes, H. L. S. and Simmons, R. G. ”Statistical probabilistic model checking with a focus on time-bounded properties.” Information and Computation, vol. 204, no. 9, 2006, pp. 1368–1409.

[5] Legay, A., Delahaye, B., and Bensalem, S. ”Statistical model checking: an overview.” In: Runtime Verification, LNCS 6418, Springer, 2010, pp. 122–135.

[6] Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J.-B., Reinkemeier, P., Sangiovanni-Vincentelli, A., Damm, W., Henzinger, T. A., and Larsen, K. G. ”Contracts for System Design.” Foundations and Trends in Electronic Design Automation, vol. 12, no. 2–3, 2018, pp. 124–400. DOI: 10.1561/1000000053.

[7] Sen, K., Viswanathan, M., and Agha, G. ”Statistical model checking of black-box probabilistic systems.” In: Computer Aided Verification, LNCS 3114, Springer, 2004, pp. 202–215.

[8] Aichernig, B. K. and Tappler, M. ”Probabilistic black-box reachability checking.” In: Runtime Verification, LNCS 10548, Springer, 2017, pp. 50–67.

[9] Musa, J. D. ”Operational profiles in software-reliability engineering.” IEEE Software, vol. 10, no. 2, 1993, pp. 14–32.

[10] Wilson, E. B. ”Probable inference, the law of succession, and statistical inference.” Journal of the American Statistical Association, vol. 22, no. 158, 1927, pp. 209–212.

[11] Clopper, C. J. and Pearson, E. S. ”The use of confidence or fiducial limits illustrated in the case of the binomial.” Biometrika, vol. 26, no. 4, 1934, pp. 404–413.

[12] Brown, L. D., Cai, T. T., and DasGupta, A. ”Interval estimation for a binomial proportion.” Statistical Science, vol. 16, no. 2, 2001, pp. 101–133.

[13] Agresti, A. and Coull, B. A. ”Approximate is better than 'exact' for interval estimation of binomial proportions.” The American Statistician, vol. 52, no. 2, 1998, pp. 119–126.

[14] Conover, W. J. Practical Nonparametric Statistics, 3rd ed. Wiley, 1999.

[15] Hollander, M. and Wolfe, D. A. Nonparametric Statistical Methods, 2nd ed. Wiley, 1999.

[16] Wilks, S. S. ”Determination of sample sizes for setting tolerance limits.” Annals of Mathematical Statistics, vol. 12, no. 1, 1941, pp. 91–96.

[17] Frankl, P. G., Hamlet, D., Littlewood, B., and Strigini, L. ”Evaluating testing methods by delivered reliability.” IEEE Transactions on Software Engineering, vol. 24, no. 8, 1998, pp. 586–601.

[18] Hamlet, D. and Taylor, R. ”Partition testing does not inspire confidence.” IEEE Transactions on Software Engineering, vol. 16, no. 12, 1990, pp. 1402–1411.

[19] Christensen, K., Varshosaz, M., and Pardo, R. ”ProbTest: Unit Testing for Probabilistic Programs.” In: Bianculli, D. and Gómez-Martínez, E. (Eds.), Software Engineering and Formal Methods (SEFM 2025), LNCS 16192, pp. 91–109. Springer Nature Switzerland, 2026. (The chapter carries a 2025 date in some indexes while the proceedings volume is dated 2026; the volume year is used here.)

[20] Agha, G. and Palmskog, K. ”A survey of statistical model checking.” ACM Transactions on Modeling and Computer Simulation, vol. 28, no. 1, 2018, pp. 1–39.

[21] The mavai project family: probabilistic testing frameworks. Source code and documentation available at https://mavai.org and https://github.com/mavai-org.