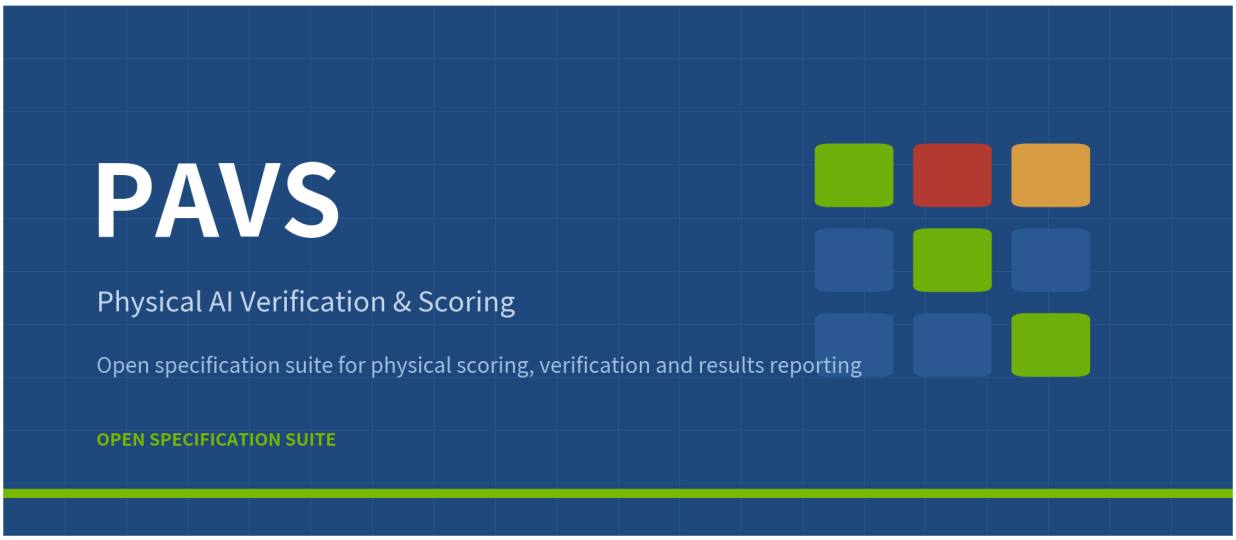


Part 2 — Casebook (informative) · English translation

Issued edition v1.3 · 29 October 2026
Published by ImmersiveCast Co., Ltd. · open specification at the proposal stage (not a standard of a formal standards body)
English translation — the Korean text is authoritative · immersivecast.com/pavs

Contents

How to read (informative)

A. Items — definition of what is measured

B. Measurement — the provenance of values

C. Verdict — pass criteria

D. Verification — scoring the scorer

E. Reporting — how results are described

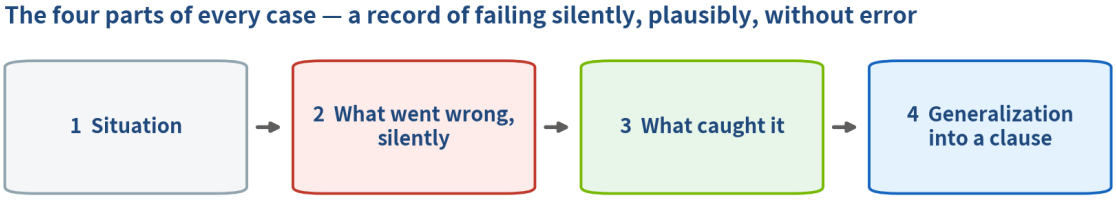
Closing remarks (informative)

Physical AI Verification & Scoring — the incidents from which the clauses were born

> **English edition — translation.** This is the English translation of the Korean edition v1.3 (2026-10-20). The Korean text is authoritative; where the two differ, the Korean text prevails. Assertion identifiers, clause numbers, case identifiers, versions, dates and figures are identical to the Korean edition.

> **Document status**: Issued edition v1.3 · 2026-10-29 · previous issue v1.2 2026-10-19 · first public release v1.0 2026-09-09 · published at immersivecast.com/pavs · target specification = PAVS Part 1 v1.5 (72 assertions · 8 ★core · requirements identical to v1.2–v1.4). **This Casebook (commentary) explains matters specified in the body and annexes of Part 1 and matters related to them, and is not part of the specification.** This document is informative (explanatory) in its entirety; requirement expressions in the case descriptions are merely summaries of Part 1 clauses, and this document creates no requirements. **Edition history**: v1.0 (2026-09-09, 33 cases) → v1.1 (2026-10-02, 38 cases · 5 new · correction of clause correspondence) → v1.2 (2026-10-19, 39 cases · 1 new · Case D14 · follow-up NOTE to D11 · notation corrections) → v1.3 (2026-10-20, 39 cases · errata correction — PAVS self-reference ‘standard’ → ‘specification’ · citations of other editions deleted · title wording of C3 and E2 · Korean term for ‘manufacturer’ aligned · 2 short names of related clauses · label in the conventions of description · Korean edition: general words replaced following the glossary of the Korean expression norm, English wording unchanged). **Standing**: Just as aviation safety regulations came out of accident investigation, the clauses of PAVS were generalized from incidents experienced in the actual operation of simulation-based robot verification. This document is that record. The cases are not a list of achievements — many of them are records of "a defect of the verification system itself, caught by a mechanism of the verification system", and applying the same criteria to itself is the target state of this specification. **Conventions of description**: Every case has a 4-part structure — ① Situation ② What went wrong, silently ③ What caught it ④ Generalization into a clause (label within the cases = ‘Generalization’). The head of each case gives the related clauses and the time of the incident (year and month). Systems, robot types and identifiers have been anonymized, and of the figures, only the measured values essential to the substance of the incident have been retained, as values at the time of the incident.

How to read (informative)



The four parts of every case — the clauses were generalized from incidents, and each case records one.

The common form of these cases is **‘going wrong silently, without any error indication, in a plausible-looking form’**. The system terminated normally, the logs held no errors, and figures were produced. Those figures were false passes or false fails. The specification's requirements for scoring, verification and reporting exist precisely because errors of this form are not caught by human attention.

The cases are arranged along the five chain groups among the 6 requirement groups of Part 1: items (A) → measurement (B) → verdict (C) → verification (D) → reporting (E). Cases of the operations (OPS) group are managed as candidates for inclusion in a revised edition. **Case**

D6 is an incident that the reference implementation of this specification found while checking itself, and one clause was newly created as a result — a record of this document's premise, that a specification is born from incidents, operating once more within the suite.

A. Items — definition of what is measured

Case A1. A speed of 0.8 passed because it was less than a force of 5.0

Related clauses: **PAVS-ITM-03** (declaration of units), **PAVS-ITM-04** (guaranteed measurement of metrics in all kinds of test)

Time of incident: 2026-08

Situation. A collaborative-safety item, "contact detection threshold ≤ 5.0 N", was returning pass verdicts.

What went wrong, silently. The value supplied to that item was not a force (N) but a speed of 0.8 m/s. Looking only at the numbers, 0.8 < 5.0, so it passes. A complete census found 17 items being scored with a physical quantity different from their own — time to collision (seconds) was scoring a contact-force (N) item, and a count of stops was scoring a response-time item. The root cause was using a clause number to carry meaning: the same clause number in a reference standard denotes different physical quantities depending on the edition.

What caught it. Each measurement was made to declare "the definition of what it measures" as a unit, and scoring was changed so that, where that unit differs from the unit the item requires, the scoring itself is refused (treated as not-measured + reason recorded). Whether this correction had made verdicts more lenient was verified separately with an adversarial sample — withdrawing a wrong verdict yields "not-measured", not a pass.

Generalization. Anchoring is done by declaration of the physical quantity, not by clause number. Scoring with a mismatched unit is not an error but **a test that never took place**.

Case A2. The name of an item did not match what it actually measured

Related clauses: **PAVS-ITM-02** (1:1 with assertions), **PAVS-ITM-05** (discriminating power by measurement)

Time of incident: 2026-08

Situation. An item "maintain clearance from obstacles" had been producing passes stably for a long time.

What went wrong, silently. It was found in the verdict-grounds path. The measured key left in the verdict-grounds string of an actual test was not clearance but **total distance traveled**. When the name and what is actually measured differ, the item becomes a document vouching for something it does not measure, and records of passing accumulate in that state.

What caught it. A procedure auditing the 1:1 correspondence of item, assertion and check. What was compared was not the item name but the substance of the check that executes.

Generalization. The substance of an item is not its name but the check it executes. Every item shall correspond 1:1 to an executable assertion, and that correspondence is itself an object of audit.

Case A3. An item that comes out only one way can turn out not to be a test

Related clauses: **PAVS-ITM-05** (discriminating power), **PAVS-ITM-06** (reverse-direction control), **PAVS-ITM-07** (corpus)

Time of incident: 2026-08

Situation. An item for compliance with a speed-restricted zone had recorded only violations across its entire proven test history — a compliance verdict had never once been produced.

What went wrong, silently. An item that produces only a single direction admits two interpretations: the possibility that every instance really was a violation, and the possibility that the gate is structurally unable to produce the opposite value. In the latter case the item does not function as a test, and the two states cannot be distinguished by outward appearance.

What caught it. A reverse-direction control — a counterpart scenario in which only one line of stimulus in the original scenario was changed to the compliant direction — was built to check whether a compliance verdict is actually produced. The design principle was left alongside it in the header of the control file: adjusting the limits of an item to produce the desired verdict is not verification but manipulation, and the controls mix violation, compliance and ratio-type items so as to discriminate even "which measuring instrument is alive when a value comes out".

Generalization. The substance of the specification is not a list of items but **an executable corpus of scenarios + controls + expected verdicts**. The boundary of an item that has produced only a single verdict is closed only with a reverse-direction control.

Case A4. Without third parties in the baseline, the same robot both passes and fails

Related clauses: **PAVS-ITM-08** (baselines with conditions)

Time of incident: 2026-08

Situation. In a sidewalk delivery task, when pedestrian density was raised slightly to 0.25 persons per 100 m, success fell from 5/5 (100% · Wilson 95% lower bound 56.6%) to 0/5 (0% · Wilson 95% upper bound 43.4%) (5 seeds per density).

What went wrong, silently. The robot's performance had not degraded. The formula for the time baseline (SLA) did not include the time spent yielding to pedestrians, so the margin, calculated for an empty sidewalk, was only a few seconds. In the same family of formulas, an omission of this type (omission of a time-cost term) had already recurred twice. During construction, model defects that produce plausible-looking figures without any error indication were also detected one after another. One example was judging contact by center-to-center distance, which misjudged nearly half of normal passing encounters as contacts.

What caught it. The principle that a baseline shall be defined together with the third-party density condition was registered: if the condition is not written down, the same robot on the same route both passes and fails.

Generalization. Performance and safety baselines shall be defined together with the conditions under which they hold (surrounding humans, environment and load). A baseline without conditions is a verdict that cannot be reproduced. (The sidewalk width in this case is based on declared values, and the test was performed on a 1D kinematic model.)

Case A5. An item was attached that, in principle, could not be measured in that test family

Related clauses: **PAVS-ITM-04** (guaranteed measurement of metrics in all kinds of test), **PAVS-VDT-01** (3-valued verdict — treatment of absence of measurement for mandatory items)

Time of incident: 2026-08

Situation. A group of civil aerial scenarios (infrastructure inspection, search and rescue, urban patrol, etc.) all came out as fails in scoring.

What went wrong, silently. A ‘no collision’ item was attached to those scenarios, but the aircraft of this family are driven by a kinematic model, so **a collision quantity is, in principle, not produced**. A complete check of 27 tests found 0 measured instances. Because the verdict chain does not fold not-measured into a pass, it produced a fail, and **that verdict itself was correct.** The defect lay not in the verdict chain but in an item being attached that could not apply to that family.

What caught it. The verdict rule that does not fold not-measured into a pass. Without this rule, a value that was never produced would have been read as ‘0 violations’ and become **a false pass**. The remedy was not to lower the threshold or fill in the value but **to remove the item from that family**. **PAVS-VDT-01** of Part 1 v1.2 made this treatment explicit — an absence of measurement of a mandatory item is counted as a fail for the reason ‘absence of measurement’, and the count of not-measured results is retained separately.

Generalization. An item presupposes that measurement is established in the test family to which it applies. An item for which measurement is not established is not an item in that family, and neither leaving it as a fail nor folding it into a pass is right — **removing it from the item list and recording that fact** is right. An honest verdict rule signals such a case with a ‘fail’, and mistaking that signal for a defect of the verdict chain leads to a wrong correction in the direction of relaxing the rule.

Case A6. Three safety clauses always passed — not because the robot was safe

Related clauses: **PAVS-ITM-05** (classification of discriminating power by measurement), **PAVS-ITM-06** (reverse-direction control)

Time of incident: 2026-08

Situation. Three safety clauses on human clearance, personal space and collision with people had never once produced a fail in the mobile-robot family.

What went wrong, silently. To check discriminating power, **a control with 10 people standing like a wall across the path** was built. The result was 0 personal-space intrusions and 0 collisions with people, but other values recorded alongside told the reason — the robot's total distance traveled was 2.04 m, and its minimum distance to a person was 3.01 m. The runtime configuration was set to **stop 3.5 m in front of a person**. The personal-space threshold is 1.2 m, so **it cannot be intruded upon in principle.** The constant pass of the three clauses was not a virtue of the robot but **a runtime guarantee**.

What caught it. The discipline of classifying an item that has never produced a fail as ‘unverified boundary’ and closing it only with a control that applies the stimulus in the reverse direction. In this case the item **was not closed** — for the test to be established, the stop distance would have to be set shorter than the threshold, and that is an unsafe configuration. It was therefore entered in the results document not as ‘tested’ but as ‘constant pass by runtime guarantee’.

Generalization. An item that always passes is one of two things — either the system under test (SUT) is excellent, or that situation **cannot be reached**. What separates the two is a

control, and passes accumulated without a control are not evidence of capability. An item for which the test is not established shall not be counted as a pass but shall be entered separately together with its reason. **The bundle of scenario, control and expected verdict is the specification in its tangible form**, and the clause text alone cannot establish this distinction.

Case A7. The metric was measuring the scenario, not the robot

Related clauses: **PAVS-ITM-05** (classification of discriminating power by measurement)

Time of incident: 2026-08

Situation. To check whether a workspace-occupancy clause has discriminating power, a test was run with the same robot and the same walking path, changing **only the zone size** from 15 m to 3 m.

What went wrong, silently. The verdict changed from 0.618 (fail) to 0.0 (pass). The robot's behavior was identical. Because the denominator of this metric was the observation window of the whole test, **the value was a property of the scenario, not of the robot**. The clause had discriminating power but did not discriminate robots.

What caught it. A procedure that did not stop the check of discriminating power at 'whether the verdict splits' but went on to **the factor that splits the verdict**. A controlled test changing only a single variable made that distinction.

Generalization. The fact that a verdict splits does not by itself show that an item is valid. A metric that splits only when the scenario configuration is changed cannot serve as a basis for comparing SUTs, and the moment the scores of different test targets are placed side by side, **a difference in scenario reads as a difference in capability.** A classification of discriminating power shall record, along with it, what is being discriminated.

B. Measurement — the provenance of values

Case B1. The top speed was not a measured value but "command × 1.35"

Related clauses: **PAVS-MEA-01** (proof of provenance), **PAVS-MEA-07** (effective value stated side by side)

Time of incident: 2026-08

Situation. In mobile-robot safety scoring, a top-speed metric was being used in the verdicts of several safety clauses.

What went wrong, silently. That value was not the speed the robot actually achieved but the drive commanded value multiplied by a coefficient. The top speed of a robot that actually ran at 1.4 m/s was recorded on paper as 4.7 m/s, leaving the safety clauses bound to that key in a state where false fails could be produced — and speeding that did not exist actually switched on a violation counter. The determination was made from 3 lines of evidence: that the value was an exact multiple of the command; that the values of two different tests were identical to the last decimal place (an agreement impossible for measured values); and that the value physically contradicted the total distance traveled. The decisive comparison was another test family — a metric of the same name was measured in that family. **The same metric name had a different provenance in each test family.**

What caught it. A displacement-based measurement providing a measured cross-check was newly established and always stated side by side with the command-derived value, and declaration of the provenance of values (measured/command/default value) was made a rule. However, the command-derived value has not yet been removed from the scoring input

and remains stated side by side with the measured value, and, as of this edition, this state is recorded as **PAVS-MEA-01** (no scoring with non-measured values) not being met.

Generalization. Scoring that does not ask about provenance goes wrong in both directions — mistaking a commanded value for a measured one produces a false fail, and substituting 0 for absence produces a false pass. The same mechanism (proof of provenance) blocks both.

Case B2. A 0 with no measuring entity was treated as a ‘pass’

Related clauses: **PAVS-MEA-02** (missing ≠ 0), **PAVS-VDT-02** (positive evidence), **PAVS-REP-03** (coverage list)

Time of incident: 2026-08

Situation. A collision count of 0 was passing a safety item.

What went wrong, silently. That 0 was a value with no measuring entity — absence substituted by 0. A sweeping survey for the same form turned up 22 instances: an entry with 0 tests was being reported as "certified" (the indication at the time), 0 records of tool use as "eligible" (the indication at the time) with an error rate of 0, and an unobserved time to collision as an "observed value of 999 seconds" — where the data type forced a number and there was no means of expressing ‘none’, the type forced a false value. There was also an instance in which the display layer recomputed, with its own formula, a rule corrected in the backend and restored the pass indication.

What caught it. An executable guard that judges ‘a pass produced from an input with no measurement values at all is a violation’. The first run of that guard also caught 2 new defects introduced in the course of the correction (including one in the over-correction direction).

Generalization. Missing data is not 0 — if a robot that made no emergency stop is recorded as having stopped in 0 seconds, it gets a full score. In addition, unless verdict rules are concentrated at a single point, the old result is restored in the display layer even after correction.

Case B3. A closed loop that trusted a broken measurement did worse than an open loop

Related clauses: **PAVS-MEA-03** (liveness), **PAVS-MEA-04** (probe with a known stimulus)

Time of incident: 2026-07

Situation. A visual correction servo was taking a measured error and fine-tuning the grasp position. It tried five times and failed five times.

What went wrong, silently. Because of an indentation error, the correction loop itself had never once executed — so the successes before that were not to the credit of this correction. When the loop was restored, the measurement value serving as the input of the correction turned out to be **constant** regardless of the trial, and the servo, believing it to be a valid error, injected the same wrong correction every time and failed every attempt.

What caught it. Verification of the liveness of measurement inputs — if the input state changes while the measurement value stays unchanged, it is treated not as a valid signal but as a measurement fault. There is one more incident of the same type. A mismatch in camera coordinate conventions made the rendering entirely invalid, yet a cross-check reading back the configured values shared the same convention and returned "normal" — determination was possible only with an independent probe that placed a known target (a single-color object) and counted the pixels of the end-to-end output.

Generalization. The soundness of a measurement is confirmed not by a self-consistency cross-check but by an end-to-end probe with a known stimulus. A closed loop that trusts a broken measurement injects error systematically.

Case B4. The scorer made the SUT slow

Related clauses: **PAVS-MEA-08** (no hard-coding of limits)

Time of incident: 2026-02

Situation. A policy trained over the full speed range showed, under test, speeds lower than its design range and unstable behavior.

What went wrong, silently. A speed ceiling was embedded as a constant in the code of the scoring system. The SUT could not use its own design range, and the score reflected not the capability of the robot but a constant of the scorer.

What caught it. A rule making every restriction derive only from the declared specifications of the SUT (the manifest).

Generalization. The scorer shall not distort the SUT. Where the test conditions differ from the design conditions of the SUT, that fact itself shall be stated in the results.

Case B5. A 3.6 km route was represented by 161 points

Related clauses: **PAVS-MEA-05** (statement of the qualification of proxy metrics)

Time of incident: 2026-08

Situation. In a sidewalk delivery task, whether a passage could be traversed was judged by a single value, the width at the end point of each segment. The route was 3.6 km, the width samples numbered 161 points, and the sample spacing was 22.0 m median and 34.6 m maximum.

What went wrong, silently. If the width at the end point was sufficient, the whole segment was judged traversable, and narrow spots between one sample and the next did not enter the verdict. In one segment the actual lateral clearance was computed as **−102 mm** — a negative value means geometrically impassable, yet the verdict was pass. The end-point width was not a physical quantity but **a function of a discretization condition, the sample spacing**, and it was used as the representative value of the segment without its qualification being declared.

What caught it. The input to the verdict was changed from the end-point width to **the minimum width of the segment**. In the same chain, a defect was also detected in which the spatial reference point was placed at the target waypoint rather than at the robot's actual arc length, so that the reference point jumped 13.9 m in one tick; this was confirmed not by conjecture but from measurement logs.

Generalization. A sample representing a segment shall be the least favorable value in that segment. A metric whose representative range is not declared is a proxy, and if the qualification of the proxy is not written down, the verdict becomes a function not of physics but of the sample spacing. In this case the same defect also exists in the preceding traversability check, but because correcting it would shift the verdicts of 14 existing scenarios along with it, it was **registered as an uncorrected gap** — recording a gap and closing a gap are different matters, and leaving it unclosed without a record is worse.

Case B6. A metric of the same name meant different things in each test family

Related clauses: **PAVS-MEA-01** (proof of provenance · no scoring of non-measured values)

Time of incident: 2026-08

Situation. In Case B1, the mobile robot's ‘top speed’ metric was confirmed to be not measured but 1.35 times the cruise command. Because a metric of the same name also existed in the legged-robot family, a blanket substitution across all families was considered.

What went wrong, silently. The same metric in the legged-robot family **was measured**. The value for that robot was 1.4985 m/s and the commanded value was 2.2 m/s, so applying the 1.35-times relation confirmed in the mobile-robot family would have required 2.97 m/s, but it was not so. **That keys had the same name was no basis for their values having the same provenance.** Had the blanket substitution been carried out, the values of the family whose provenance was correct would have been replaced by commanded values, and a defect would have been injected where there had been none.

What caught it. A procedure confirming the provenance of values individually for each family before substitution. The determination looked at three things together — whether the value is an exact multiple of the command, whether it is identical between runs down to the decimal places, and whether it contradicts the total distance traveled.

Generalization. The provenance of a value belongs not to the name of the metric but to the chain that produced the value. A blanket correction of metrics with the same name shall be carried out only after provenance has been confirmed for each chain. The finding of a defect applies only to the scope in which that defect was found, and extending it without confirming the scope is not a correction but a new defect.

Case B7. The reflected value was correct, but there was no effect

Related clauses: **PAVS-MEA-07** (requested value, effective value and application stated side by side), **PAVS-MEA-04** (end-to-end probe with a known stimulus), **PAVS-VRF-03** (confirmation of effective values)

Time of incident: 2026-10

Situation. The rated-payload condition required by an international standard was added to a robot-arm position accuracy test. The test load was applied by adding the rated payload to the mass of the arm's end link, and whether it had been applied was to be confirmed by reading back the mass. In the first pilot run, the mass read back was +10 kg, as requested.

What went wrong, silently. After the load was added, the displacement of the arm's end, settled to the same joint target, was 0.000 mm. The arm with 10 kg added did not sag at all. The physics engine puts an articulation at rest into a sleep state, and a change of the mass property does not wake it from sleep. The configured value was stored but was not reflected in the physics. Had application been confirmed by read-back alone, this run would have been recorded as ‘load applied’ while presenting no-load measurement values as the results of the load condition.

What caught it. An effect guard — a check that terminates the run and leaves it as not-measured if, at the same joint target before and after the load is applied, the sag of the arm's end does not exceed a threshold (0.1 μm) — fired in the first pilot run. Instead of fixing the cause by conjecture, discriminating measurements (displacement in the un-woken state, displacement after waking by rewriting the joint velocities to 0, the no-load steady-state joint error, and the sleep threshold) were added and the test was run once more. Displacement was 0 before waking and a sag of about 1 mm appeared after waking, confirming the sleep hypothesis, and the basis for judging the effect was changed to the displacement after waking. Whether the same mechanism had distorted the settling interval of the existing accuracy measurements was also checked offline using the recorded trajectories, and the estimated distortion was about 1% or less relative to the worst-case accuracy. The defect was caught before the results were issued.

Generalization. When the requested value and the effective value are stated side by side, the effective value shall be not a read-back of the configured value but **a read-back of the result of application**. A read-back of the configured value only shows that the value was stored; it does not show that the physics changed in accordance with that value — a confirmation made along the same path as the configuration shares the defects of the

configuration path (the same structure as the self-consistency cross-check of Case B3). The effectiveness of a test condition shall be confirmed end to end through the physical change that the condition has to produce (sag, in the case of a load), and a run under a condition whose effect has not been confirmed is not a result of applying the condition but is not-measured. The same discipline applies to the independent variable of a comparison test (Case D3).

C. Verdict — pass criteria

Case C1. A default value was returning a pass — and 89% of all 401 passes (a census)

Related clauses: **PAVS-VDT-02** (positive evidence), **PAVS-VDT-07** (retrospective audit)

Time of incident: 2026-05

Situation. A mission-success verdict had been showing a high pass rate for a long time.

What went wrong, silently. The default path of the verdict logic was returning success even when the required conditions (all waypoints reached, measured lift of the object) were not met.

What caught it. The logic was corrected so that a pass is produced only from the positive evidence "all conditions met by measurement" — and, more importantly: **every existing pass from the defect period was retrospectively audited against the new criterion.** The result, in a census of all 401 passes from the defect period: 89% were false passes (retrospective audit 2026-05). Without the obligation of retrospective audit, this figure would not have existed, and the false passes would have remained as a record of achievement.

Generalization. Correcting a defect in a verdict criterion does not end with a code correction — the correction extends to the retrospective audit of the past verdicts that the criterion produced and the disclosure of the results.

Case C2. ‘Grasp success’ was produced with no gripper attached

Related clauses: **PAVS-VDT-03** (physical cross-validation)

Time of incident: 2026-05 (contact-force determination in a similar family 2026-07)

Situation. In a pick-and-place test, grasp success, placement success and mission success were all counted, and the screen display also read "grasp complete".

What went wrong, silently. No gripper was attached, so every open/close command was a no-op. The success metric was counting the issuing of commands, not physical events.

What caught it. A rule cross-validating the success metric against physical evidence — measured lift of the object, contact force and video. In a similar family, a "success" obtained by logically attaching the object to the hand (weld) was determined by measured contact force (about 20 N for a normal grasp vs. about 0 N for a weld) and downgraded to a hard fail.

Generalization. A success metric shall be cross-validated against independent evidence of the physical event it presupposes. That a command was issued and that the world changed are different propositions.

Case C3. A large surface ship sank, yet the score was a ‘pass’

Related clauses: **PAVS-ITM-05** (discriminating power), **PAVS-REP-12** (safety anomaly ≠ ‘normal’) — coverage of the scoring sheet

Time of incident: 2026-08

Situation. In a re-verification of a maritime scenario, a large surface ship sank at constant velocity right after the start and descended through the seabed. Even so, the overall verdict of that test was a pass.

What went wrong, silently. The scoring sheet had no item asking about capsizing or sinking. **A failure that is not on the scoring sheet passes silently.** This defect is a problem not of capability but of item coverage, and the more normal tests there are, the longer it stays hidden.

What caught it. A procedure of re-testing the entire body of past passing results against improved scoring criteria brought this incident to light. Re-verification classifies the results as follows: those whose verdict was confirmed to stand, passes that had no verdict grounds at all (not-measured), verdicts that had gone stale, and undetected defects revealed by the new criteria — while recording the attribution separately so that a change in the measuring instrument is not misattributed to a regression in capability. The sinking and capsizing item whose necessity this incident revealed has, as of this edition (2026-10), not yet been implemented as a verdict item — the gap is recorded but not closed.

Generalization. A scoring system shall periodically re-test its own coverage. "Failure modes that were unknown at the time" are caught only by creating new items and by retrospective re-verification.

Case C4. Same robot, same task — change only the specifications and the verdict is reversed

Related clauses: **PAVS-VRF-05** (model authenticity), **PAVS-VDT-08** (rejection of unperformable directives)

Time of incident: 2026-08

Situation. The same task of the same robot arm was scored separately with the specifications from a public model repository and with the manufacturer's confirmed specifications.

What went wrong, silently. In the public specifications, the wrist joint limit was registered as wider than on the real robot. With those specifications, a task in a pose that the real robot cannot physically assume was produced as a **pass** (false pass). Other axes of the same model were registered as narrower than on the real robot, causing attainable poses to be lost, and in measurement a defect in the gripper actuation specifications made a task with a load within the rating fail (false fail). Correcting only the figures in the specification file did not change the behavior. The point at which the physics engine actually enforces the limits lay elsewhere.

What caught it. Verification by comparison with the manufacturer's confirmed specifications, and the rule that rejects the test itself rather than silently approximating a directive that cannot be performed (fail-loud) — under the confirmed specifications, the task was handled not as a "pass" but as "test rejected".

Generalization. In a system that scores third parties, model authenticity is a precondition of scoring. Unless the authenticity of a brought-in model is verified first, even a high score proves nothing.

Case C5. Pass-bys while the robot stood still were counted as the robot's contacts

Related clauses: **PAVS-VRF-04** (disproof before attribution to the SUT)

Time of incident: 2026-08

Situation. In a sidewalk delivery task, the number of contacts with pedestrians and bicycles was counted. Events in which a pedestrian passed beside the robot while the robot stood waiting in a narrow section were classified as contacts.

What went wrong, silently. In a section with a passage width of 762 mm, 5 contacts were counted. The robot was stationary at those moments and could perform none of avoidance, deceleration or route change. **The events were real but were not acts of the SUT**, and the verdict was a false fail. Pass-bys in which another party overtook from behind were classified in the same way.

What caught it. The classification criterion was changed to ‘the party that was able to act at that moment’. Pass-bys that the SUT could not act upon were excluded from contacts but **counted in a separate category (sidewalk occupancy)**, so that the event itself remains in the score but does not enter the SUT's verdict.

Generalization. The occurrence of an event and the attribution of an event are different. Before an event is attributed to the SUT, it shall be confirmed whether the SUT was able to act with respect to that event. Attributing an event with no possibility of action produces a false fail, and deleting the event is a loss of information, so it shall be preserved in a separate category. **PAVS-VRF-04** limits the object of disproof before attribution to the soundness of the measurement chain, and this case suggests that the object **has to be extended to the possibility of action** — it was registered, as a candidate for clause extension, among the matters for review at the next formal revision under 12.6 of Part 1.

Case C6. Body contact outside the code table slipped through as a pass

Related clauses: **PAVS-ITM-04** (guaranteed measurement of metrics in all kinds of test), **PAVS-VDT-02** (positive evidence), **PAVS-VRF-05** (model authenticity — comparison with the declaration), **PAVS-VDT-07** (retrospective audit)

Time of incident: 2026-09

Situation. In a robot-arm grasping test, contact in which the gripper body (palm) presses on the object was counted as a collision, and contact in which a finger touches the object was exempted as intended grasping. When five robot types were tested on the same task, only one robot type recorded 0 collisions in all 25 runs, becoming the only one of the five to pass the collision item of the verdict chain.

What went wrong, silently. The classifier that separates the roles of contacts (palm, finger, arm) referred only to a list of palm part names embedded in the code. The name of that robot type's gripper body part was not on the list, and it was classified as a finger because the part path contained the gripper product name. Contact between a finger and the object is exempted as intended grasping, so none of the 25 runs were counted. The measured contact force was of the same magnitude as palm contact on the other robot types and about 10 times the finger grasping force. The robot declaration for that robot type had specified the part as the body since 2026-09-06, and another measurement in the same system (descent target limit) was reading that declaration. One fact was written down separately in two places, and only the classifier side did not look at the declaration. ‘0 collisions’ was a result not of safety but of the classification rule.

What caught it. A procedure that did not read the unresolved question ‘0 collisions for one robot type only’ as evidence of safety but checked it along three axes: contact force, the hierarchical position of the part, and the robot declaration. All three axes pointed to ‘that part is the palm’. The classifier was fixed to read the robot declaration, but declared names were accepted only within the gripper substructure — because the same name is also commonly used for arm base parts, and going by the name alone would classify the arm base as a palm. After this was pinned down with counterexample tests (the arm base is

classified as arm even with a declaration; robot types without a declaration show no regression with the existing list), the baseline was frozen again. In the re-freeze, the contacts of that robot type were counted in all 25 runs, false detections counting a grasp as a collision were 0 across all robot types, and the strict success metric was unchanged for all robot types. Right after the re-freeze, it came to light that a tool that counts part of the verdicts by offline replay had not been receiving the same declaration. The tool was fixed the following day, but re-judging of the frozen verdicts that the tool had already produced was omitted; this was discovered the day after that through a comparison made during other work, and the verdict concerned was corrected from pass to a grade one step lower.

Generalization. When a verdict rule refers to a ‘table written in the code’, a real object that is not in the table falls into the default classification without error, and if that default classification is an exemption, the result is a false pass. The basis of classification shall be the declaration of the SUT, not a list embedded in the code, and when the declaration is read, the scope within which that declaration is valid (in this case, the gripper substructure) shall be restricted along with it. Where several consumers use one fact, after the authoritative source is moved it is confirmed that all consumers have followed, and existing verdicts produced by consumers that did not follow are re-judged retrospectively. Without counterexample tests that include real objects outside the table, this defect hides behind the best possible score, ‘0 instances’.

Case C7. The value was there, but the proof of measurement was not

Related clauses: **PAVS-VDT-01** (3-valued verdict — treatment of absence of measurement for mandatory items), **PAVS-MEA-01** (proof of provenance · no scoring of non-measured values)

Time of incident: 2026-09

Situation. In a test in which 3 small bipedal walking robots patrol indoors, the person in charge of the test compared the values of 7 items against their thresholds by hand and reported ‘7/7 passed’. The formal verdict chain judged the same run as 4 passes and 5 fails out of 9 items.

What went wrong, silently. Of the 5 fails, 1 was an actual collision, and 4 had the same reason — ‘This metric has a value, but there is no record that it was measured in this run. The existence of a value is not evidence of measurement, and since this is a mandatory item, fail’. The cause lay in three paths that merge measurement results into the results file (per-robot and inter-robot relational metrics, contact-monitoring metrics, sensor-record counts) loading only the values without recording them in the measurement-proof ledger. The measurements actually ran, but that fact was not proven. Because this was the first test to attach items to multiple robots and to contact-monitoring metrics, it had not surfaced until then. The human comparison read the fact that a value existed as the fact that it had been measured.

What caught it. The formal verdict chain, which requires proof of measurement rather than a value and leaves a reason on a fail. The treatment permitted by **PAVS-VDT-01** of Part 1 v1.2 — counting an absence of measurement of a mandatory item as a fail for the reason ‘absence of measurement’ — worked exactly as specified. Because the reason was identified, these fails could be attributed not to a failure of the SUT but to a defect in the measurement wiring. All three paths were fixed to declare the measurement at the point where the measurement executes, and a guard was newly established — if an item uses a metric that ‘has a value but is not in the ledger’, the guard fails that run, while distinguishing the case where the measuring instrument did not run (no value at all) from a wiring defect (value present, no proof). After the first path was fixed, this guard found the remaining two paths. Comparing the proof ledgers of the same scenario before and after the correction

confirmed that the metrics concerned are recorded as ‘measured’, and the full regression over all examples passed.

Generalization. The input to a verdict is not the existence of a value but the proof of measurement. Comparing a value with a threshold alone does not establish a verdict, and a manual human comparison that does not look at the proof of measurement goes wrong in the lenient direction. A fail of a value without proof shall carry a reason — without a reason, a failure of the SUT and a defect in the measurement wiring look like the same ‘fail’, and with a reason, the place to fix is pinpointed. The rule that does not fold not-measured into a pass (Case B2) and the rule that identifies the reason for a fail are useful only when they exist together.

D. Verification — scoring the scorer

Case D1. The scorer itself takes the test

Related clauses: **PAVS-VRF-01** (negative control)

Time of incident: 2026-07

Situation. A group of items scoring compliance with rules of engagement was operating stably — the scorer's ability to detect violations, however, is a separate object of verification.

What could have gone wrong, silently. This case is the record not of an accident but of a preventive mechanism — without verification, a regression that makes the scorer lenient passes undetected through a suite made up only of passes.

What caught it. A scenario that violates deliberately (engagement without authorization) was placed permanently in the regression suite. The suite as a whole was set to pass only when this scenario is judged a fail — with the violated item remaining in the verdict grounds — and, as of 2026-08, the automatic comparison of the expected verdict against the actual verdict was performed every time the suite was run. The wiring that runs that suite automatically on a periodic basis, however, was lacking (**PAVS-VRF-16**, Case D6).

Generalization. The discriminating power of a scorer is an object of standing tests, not of assertion. A regression suite containing only passes verifies only half of the scorer.

Case D2. The re-measurement discipline detected a regression by the discipline's own author

Related clauses: **PAVS-VRF-02** (golden set), **PAVS-VRF-11** (change impact)

Time of incident: 2026-08

Situation. A seemingly trivial change of sort order went into the measurement code, together with the judgment that "order does not change the verdict".

What went wrong, silently. That judgment was a conclusion drawn from the form of the condition alone. In reality the length of the measurement path changed, and the accuracy measurement value of one robot type worsened by a factor of 15.

What caught it. The discipline that "when measurement code changes, every robot type is re-measured and compared against the snapshot". The re-measurement exposed the regression immediately, and after the revert, bit-level reproduction confirmed that the restoration was complete. A lesson was registered alongside it: **a judgment that something is "unrelated" is made only after confirming in the code "what it is a function of".**

Generalization. A change to scoring or measurement code shall be accompanied by re-evaluation of the existing verdicts that use that code as their denominator. The verification system applies the same discipline to changes in itself.

Case D3. The independent variable of the disproof experiment had never changed, in either group

Related clauses: **PAVS-VRF-03** (confirmation of effective values)

Time of incident: 2026-07

Situation. An A/B experiment verifying the effect of the physics update period ended in "no difference — hypothesis disproved", and operating rules were even built on that conclusion.

What went wrong, silently. A separate guard defect was downgrading the effective period of both test groups to the same value. It was an experiment in which the independent variable had never once changed — not a disproof but a void result. The contamination persisted for several weeks.

What caught it. The rule that the independent variable of a comparison test is confirmed in each group individually, from the effective value in the log rather than from the configured intent.

Generalization. A comparison test whose effective values have not been confirmed is not a disproof but unverified. A wrong disproof produces a wrong rule, and that rule stays in place for a long time.

Case D4. A measurement bug masqueraded as a robot defect

Related clauses: **PAVS-VRF-04** (disproof of the measuring instrument first), **PAVS-OPS-02** (preservation on correction)

Time of incident: 2026-07

Situation. As grasp failures recurred, they were judged to be "a geometric defect in the gripper asset", and matters came to the brink of an order to rebuild the asset.

What went wrong, silently. The actual root cause was two measurement and mapping bugs — a misapplied gap conversion formula made the gripper overtravel, and the grip-force measurement overwrote its value with 0 on unrelated contact callbacks, rejecting genuine grasps as "jams".

What caught it. The procedure of disproving the measurement chain before attribution to the SUT — an independent cross-check of the gap conversion and a control test of the grip-force measurement — exposed the two bugs, and after the measurement was fixed the same asset passed in consecutive runs. The original verdict was withdrawn on the data, and the record of the withdrawal was not deleted but preserved with a link to the correction.

Generalization. Before a test failure is attributed to the SUT, the measurement chain is disproved first. A measurement defect tends to masquerade as an SUT defect — and because the history of a misdiagnosis is itself a verification asset, it is not deleted but preserved with a link to the correction.

Case D5. The safety guard had never once fired

Related clauses: **PAVS-VRF-13** (verification of guard firing)

Time of incident: 2026-08

Situation. A robot arm was being tested in an **isolated experimental cell** for the company's own research (not a collaborative environment shared with workers). This cell had no hardware emergency stop, so software was the core of its safety, and a contact-detection guard was operating "switched on".

What went wrong, silently. The path by which the guard read its baseline was a call that did not fill in the torque values, so the guard was aborting with an internal error every time — that is, the contact stop **had never once operated** throughout the period of operation. That it was switched on and that it operated were different propositions.

What caught it. After the fix, the threshold was lowered to confirm actual firing by an injection test, and a placebo control (0 false firings under conditions in which it shall not fire) was verified as well. An adversarial review caught a further pitfall: a fixed baseline can mistake gravity for contact depending on the pose — sensitivity was not a constant but a function of pose.

Generalization. Verification of a safety device extends not only to "is it called?" but to "when called, does it actually operate?": an injection test confirms firing, and a placebo control confirms false firing and insensitivity.

Case D6. The checkers existed but were not run

Related clauses: **PAVS-VRF-16** (standing execution of self-checks), **PAVS-VRF-11** (change impact analysis)

Time of incident: 2026-08

Situation. The scoring system was equipped with self-checking mechanisms: a probe that injects empty input into the verdict surface to confirm whether a pass is produced, a detector that points out stale boundary notes, and a tool that recomputes the discriminating power of items. Each of them had passed even its own self-verification (self-test), and when a defect was injected, each actually fired.

What went wrong, silently. Those tools **were not connected to any automatic execution point.** No periodic execution; no executing party in the integration checks. The sole automatic execution point was a single hook at the time of change, and that hook was gated by path scope, so it excluded from checking most of the verdict surface that the probe protects. That is, **the automatic coverage of the guard was inversely correlated with the risk surface the guard was meant to protect.** The ‘0 violations’ produced in this state was not the result of conformance but the result of non-execution, and the two states were displayed identically in the report.

What caught it. A self-inspection of conformance to the specification. Once the criterion became not whether something was implemented but **whether it was actually executed**, existence and operation were told apart. This criterion is hard for the party that developed the tools to apply to itself, because to the developing party the existence of a mechanism is perceived as its operation.

Generalization. The existence of a checking mechanism is not evidence that the check was performed. Self-checks shall run automatically, either periodically or at the time of change, and where the execution scope does not cover the whole of what is protected, the excluded scope shall be disclosed. ‘No violations’ from a check that was not run is not the result of conformance but the result of non-execution.

Case D7. The manufacturer was publishing two values for the same entry

Related clauses: **PAVS-VRF-05** (model authenticity)

Time of incident: 2026-08

Situation. A tool was being built to automatically compare the specifications of simulation models with the manufacturer's specifications. On one robot arm it caught joint limits entered narrower than the manufacturer's published values on 6 of 7 joints, with a maximum difference of 9 degrees. Being on the narrow side, this does not create false passes, but it under-reports capability, so it was registered as a candidate defect.

What went wrong, silently. The premise collapsed while a third robot type was being compared by the same method. That manufacturer's official distribution material stated the limit of one joint as **half of the hardware limit**, while recording that fact and its reason in

the same material. An adjacent joint interferes so that the joint physically cannot rotate any further, and the reduction was made to keep motion planning from failing in a discontinuous solution space.

That is, the ‘manufacturer's confirmed value’ was not a single value. The hardware limit and the description-file value reduced for planning both come from the same manufacturer. Depending on which is taken as the baseline, the same model becomes either “a defect 180 degrees too narrow” or “an exact match”.

A comparison that looks only at the equality of values judges these two robot types **wrongly, in opposite directions** — a reduction with a recorded basis as a defect, and a reduction without a basis as “just a little narrow”.

What caught it. Widening the set of comparison targets caught it. Up to the second robot type, “our values are narrow” looked like a consistent defect; only at the third did it become clear that this interpretation depends on the choice of baseline. The difference between the two robot types was not the size of the reduction but **the presence or absence of a basis** — for one, the manufacturer had written down the reason; for the other, no material contained any wording that would produce that reduction.

Generalization. What a model authenticity check shall examine is not the equality of values but **whether differences can be explained**. Where a manufacturer publishes several values for the same entry, the adopted baseline and its reason are declared, and a basis is recorded for each entry in which the model differs from the baseline. **Only a difference with no recorded basis is unverified.** The direction of a difference (wider/narrower) carries a different degree of risk and is therefore recorded with it.

Case D8. The loader's catch-all rule overrode the joint limit declared by the asset by a factor of 428

Related clauses: **PAVS-VRF-05** (model authenticity comparison)

Time of incident: 2026-07

Situation. A humanoid robot fell repeatedly during testing. In the course of looking for the cause, the actuation limits of 14 joints in the hand were checked.

What went wrong, silently. The value authored in the asset file was 0.7 N·m, but the value actually loaded into the simulation was 300 N·m — **a factor of 428**. It was the result of a catch-all rule in the robot registry (apply 300 if a joint name ends in a particular string) overwriting the asset's declaration. No error occurred, and the robot loaded normally and carried out the test. Whereas Case D7 dealt with **a manufacturer publishing two values**, this case is **one in which the asset had only one value, yet our own loading chain put in a different value**.

What caught it. A comparison of the value loaded into the simulation against the value declared by the asset. Without this comparison, the test is performed not on the ‘declared robot’ but on the ‘loaded robot’, while the results document is written in the name of the former.

Generalization. Threats to model authenticity come not only from inconsistency among external sources but also from **silent overwriting within one's own loading chain**. A catch-all rule that applies default values shall not be able to win over an explicit declaration, and the comparison of loaded values against declared values shall be a condition for starting a test. In this case, however, the fix **did not change the symptom** — the physical response of the joints was identical to the decimal place before and after the fix. Fixing a defect is not in itself identifying the cause, and at the point where two fixes have failed to change the symptom, it is the approach, not the hypothesis, that shall be changed.

Case D9. A fix for one robot type brought down two robot types that were already working

Related clauses: **PAVS-VRF-11** (change impact analysis · re-evaluation of existing verdicts)

Time of incident: 2026-07

Situation. A sensor origin offset was found to be missing from a humanoid's terrain-sensing input, and a fix compensating for it was applied. The fix itself was verified on that robot type.

What went wrong, silently. With that fix, 2 other legged robot types sharing the same sensing chain (1 quadruped and 1 biped) **had their gait collapse**. Both robot types had been operating normally until the fix. The fix had been verified by looking at the target robot type alone, and the verdicts of other robot types consuming the same chain were not re-evaluated.

What caught it. The fix was reverted in full, and the fact of the revert and its reason were registered in that chain's documentation as ‘a record to be read together with any future attempt at this fix’.

Generalization. A fix to a shared chain **moves the verdicts of every robot type that consumes the chain along with it**. Verification on one robot type is not verification of the change but a sample of the change. In a scoring system that claims to be robot-agnostic, this property is at once a strength and the largest regression path, and a change to a shared chain shall be preceded by re-evaluation of every consuming robot type.

Case D10. A measurement ‘view’ changed the physics of the SUT

Related clauses: **PAVS-VRF-02** (verdict preservation of the golden set), **PAVS-VRF-11** (change impact analysis), **PAVS-MEA-06** (confirmation of actual output from new channels)

Time of incident: 2026-09

Situation. To measure the contact by which the gripper body (palm) presses on an object, a measurement ‘view’ that reads physics state was added to the palm part. In the full re-measurement the next day, only the insertion test cell (a test unit consisting of one combination of robot type and task) of one robot type differed from the baseline — while descending to the insertion position, the end of the arm oscillated periodically by ±4 cm from side to side, after which the insertion succeeded. Looking only at the strict success metric, nothing had changed.

What went wrong, silently. The convenience function that creates the view also performed, where the target part had no rigid-body physics properties, the action of assigning those properties. That robot type's hand part had been merged into its parent part through a fixed joint when the model was imported, so it had no rigid-body properties, and the moment the view was created it became a separate rigid body. The measuring device had changed the physics of the SUT (the same family as Case B4, ‘The scorer made the SUT slow’). The results of the other robot types, whose views were created on parts that already had rigid-body properties, were byte-for-byte identical — the cause was not the creation of the view but the assignment of physics properties that had not existed.

What caught it. The monitoring tool held the discipline that the re-measurement of a test cell with a fixed object pose shall be a replica of the baseline (Case D11). When the one test cell whose replication had broken was run once more with only the view switched off, the oscillation disappeared (2 trajectory reversals/1.38 m, identical to the baseline), and a confirmation run with the view changed to read only pose was byte-for-byte identical to the run with the view switched off. A self-test was put in place to prevent recurrence of the same defect.

Generalization. Measurement shall not change the physics of the SUT — read-only access is the default, and convenience functions that 'assign the property if absent' are not used on the SUT. After measurement is added, confirming **whether re-measurement of the same input is a replica of the previous result** (or, for a test cell in which replication does not hold, whether it lies within the known range of variation; Case D14) is the completion condition of a measurement change. Confirmation of actual output from new measurement (**PAVS-MEA-06**), golden set comparison after a change (**PAVS-VRF-02**) and change impact analysis (**PAVS-VRF-11**) partly cover this confirmation, but no clause explicitly states the requirement itself that 'measurement shall not change the SUT' — like Case C5, it has been registered, as a candidate for clause extension, among the matters for review at the next formal revision under 12.6 of Part 1.

Case D11. Repeated runs of a fixed-pose test were not samples but replicas

Related clauses: **PAVS-REP-06** (statistical discipline — number of trials and uncertainty), **PAVS-ITM-09** (variation along perturbation axes), **PAVS-VRF-14** (reproduction · configuration pinned), **PAVS-VDT-01** (3-valued verdict — keeping the 'undecided' state)

Time of incident: 2026-09

Situation. There was a test cell (a test unit consisting of one combination of robot type and task) in which a single run yielded so few trajectory reversal events that the confidence interval straddled the threshold and the cell remained 'undecided' (a state in which neither a pass nor a fail can be given). To increase the sample, the plan was to run the same test cell four more times and pool the results.

What went wrong, silently. In that test cell the object pose was fixed and the physics computation was deterministic, so the trajectory records of the repeated runs were **byte-for-byte identical** even when executed on different compute nodes (within 39 µm even for test cells involving contact). Had the same trajectory been counted five times to narrow the event-count-based confidence interval, the statement 'the sample has increased' would have been false.

What caught it. A rule was put into the pooling tool to 'treat a run as a replica and exclude it from pooling if the point-wise deviation of its trajectory is within 0.1 mm'; all eleven repeated runs were excluded, and the 'undecided' verdict was kept as it was. The way to increase information was not repetition but **measuring more motion within one run** (including the accuracy test segment in the trajectory record), and when that was done, the same test cell received a verdict from a single run (13 reversals/6.27 m, confidence interval below the threshold).

Generalization. A number of repetitions is a sample only where there is independence between runs. In a simulation with perfect reproduction, the phrase 'increase the sample size' holds only once it has first been stated what changes from run to run (the random seed and the range of variation of the object pose). Replicas are recorded not as samples but only as **evidence of reproducibility**. Not filling an undecidable state with replicas to make it decidable shares the intent of the 3-valued verdict, which does not fold not-measured into pass or fail.

Follow-up (2026-10). The test cell of this case was indeed byte-for-byte identical. When fixed-pose re-runs under the same conditions (the same code and the same configuration) were later gathered again, the groups whose trajectories were identical point for point were 8 of 39, and in one test cell even the verdict diverged (Case D14). The generalization of this case concerns 'the case of replicas'; whether runs are replicas is not assumed but confirmed by comparison for every run.

Case D12. Precision was measured without the arm's own weight

Related clauses: **PAVS-VRF-05** (model authenticity comparison), **PAVS-MEA-07** (requested value, effective value and application stated side by side), **PAVS-VRF-11** (change impact analysis · re-evaluation of existing verdicts), **PAVS-VRF-02** (verdict preservation of the golden set)

Time of incident: 2026-10

Situation. The simulation model of an industrial robot arm had been measured in four tests — position accuracy, grasping, placement and insertion — and frozen as a baseline. This model had been authored by combining a gripper with a publicly distributed original model. Every run ended normally without error.

What went wrong, silently. All 9 arm links of the original model had been authored with gravity disabled, and the combined model inherited that setting unchanged. Gravity acted on the gripper and the object, but the arm itself had no weight. The position accuracy contained no sag due to the arm's own weight. Specification values such as mass and joint limits looked normal, and nowhere in the system was there a step that recorded or checked whether gravity was actually applied to each link.

What caught it. When a rated-payload test condition (Case B7) was added, the gravity of the link to which the payload was to be attached was switched off, so adding the payload could produce no sag, and the test, rather than silently approximating, refused to run (fail-loud). Tracing that refusal to its cause and reading the model revealed the defect across all the arm links. Four actions were taken — normalization of the gravity setting of the combined model (confirmed by comparison that, of the 2,241 authored properties and relationships, the only changes were the 9 gravity-disabled flags), insertion of the normalization into the authoring stage to block inheritance on re-authoring, a new instrument that records the gravity state of each link at the start of a run and reads the model file before a run is submitted, and re-measurement of the four tests concerned. In the re-measurement the verdicts of all four were maintained, but the figures shifted. When other public models were read with the same instrument, the same defect was found in another model as well, and the party responsible for that model was notified.

Generalization. A model authenticity check does not end with the comparison of specification values. It shall extend to whether physics is actually applied to those specifications — whether gravity, colliders and actuation limits work inside the simulation (whereas Cases C4, D7 and D8 dealt with inconsistent values, this case is one in which the values were right and the physics was switched off). That verdicts were maintained in re-measurement is only an after-the-fact confirmation that the defect did not change the verdicts this time; it does not make the measurements of the defect period valid. Limiting the scope of the normalization change by comparison, so as to show that no other property changed, is the minimum unit of change impact analysis. This defect surfaced at the moment a new test condition first depended on that physical quantity — a refusal issued by a new condition shall be traced not as something to bypass but as a signal pointing to a hidden defect in the model.

Case D13. The pre-registration gate was called after the run had finished

Related clauses: **PAVS-VRF-16** (standing execution of self-checks), **PAVS-OPS-06** (gate = measured basis · visible notification of rejection), **PAVS-VRF-06** (agreement = object of measurement · adoption through held-out tests), **PAVS-OPS-02** (registry entry lifecycle · preservation on correction)

Time of incident: 2026-09

Situation. Comparison experiments were set up so that the hypothesis, discriminator, random seeds and run order were fixed in a pre-registration file before execution, and a checking gate confirmed the format rules of that file. Three rules — the test-target notation format, the explicit seed list and the run-order declaration — were introduced on 2026-09-06.

What went wrong, silently. A pre-registration file registered on 2026-09-16 was run while violating all three rules, consuming 340 test runs over two days. That file had never passed through the gate. The gate was called only at the stage of writing the hypothesis verdict table after the batch had finished, and there was no validation before run submission. The initial diagnosis regarded this 1 case as a ‘record predating the rules’ and sought to decide whether to grant a grace period (exemption from retroactive application), but when the history was compared, the rules preceded the file by 10 days. It was not a question of grace but of non-enforcement. At the same time, the workspace still held an unfinalized edit that changed that file's discriminator after the results had come out.

What caught it. The procedure of comparing, through the history, the time a rule was introduced against the time a file was registered, before treating 1 exception under a grace period. The gate was moved ahead of run submission so that, on a violation, the batch is rejected with 0 runs executed. Where a run has to proceed even though the violation is known, a reason of at least 8 characters shall be left, and that reason is recorded together in the run configuration — silent bypass is not permitted. The order itself, in which the gate call precedes the writing of the hypothesis verdict table, was pinned by a test. The violating file was restored to its finalized original and not corrected — correcting an old record to fit a new rule erases what was registered at the time.

Generalization. That a check is wired in and that a check is enforced are different things. If Case D6 was ‘a check connected to nothing’, this case is **a check that is connected but in the wrong place** — a pre-check called after the run cannot prevent a violation and only records it after the fact. The validity of a check is judged not by its existence but by its call timing — whether it is called before the run it is meant to protect — and that call order shall be pinned by a test. A grace period for records predating a rule and non-enforcement after a rule look the same on the surface, so they are told apart by comparing timing. Correcting a pre-registration after seeing the results nullifies the pre-registration; the old record is preserved as it is, in its violating state, and the correction is linked as a separate record.

Case D14. Re-runs of the same pose changed the verdict

Related clauses: **PAVS-VRF-14** (reproduction · configuration pinned), **PAVS-VRF-09** (deterministic reproduction treated first as a system defect), **PAVS-REP-06** (statistical discipline — number of trials and uncertainty)

Time of incident: 2026-10

Situation. In one insertion test cell with a fixed object pose, the results of runs executed with the same code, the same configuration and the same pose split into two paths. In one path the part slid straight in at the hole entrance; in the other it jammed at the entrance and went in only after nine rounds of jam search. Both paths succeeded in the insertion, but the back-and-forth of the search was counted as trajectory reversals, so the test cell verdict split between ‘pass’ and ‘caution’. The baseline was reusing one reference-pose run measured earlier, and that one run was the good path.

What went wrong, silently. The repeated runs of this test cell had been assumed to be replicas (Case D11). Under that assumption there was no reason to re-measure the reference pose in each round, and the one reused run was fixed in the baseline as what was in effect a draw of one of the two paths. When 11 reference-pose runs from code versions producing the same physical trajectory were gathered, 3 were the bad path. Among themselves, the bad-path runs had trajectories identical point for point even across runs of different code versions on different days — not random noise, but two deterministic paths.

What caught it. Without blaming the split on the environment or on concurrent runs (**PAVS-VRF-09**), the branch point was narrowed down by solo sequential runs. When measurement recording joint and object states bit for bit was added, the two paths had identical states right up until the part held in the hand was observed by the camera, and only the fingerprint of the depth image used for that observation differed. The depth image was computed differently in the last digit from one run (process) to another, so the observed value of the part tip differed by 22 nm, and alignment and first contact turned this difference into the branch between sliding in and jamming. Causality was established by a control that swaps only the observation values — overwriting just the observation bits of the bad path onto a run that had produced a good observation reproduced the bad path exactly, down to the trajectory. Because re-capturing the same scene ten times within the same run gave the same depth image every time, the split was determined per run, not per frame. Reproduction by recomputing verdicts from stored outputs (**PAVS-VRF-14**) held regardless of this split — what diverged was not the scoring but the re-running of the simulation. As a treatment, the method of rounding the observation values to fix the result on one side was not used, because if the real robot has the same sensitivity, the score would come out better than the real robot's. The behavior of the test cell was left as it was, the reuse of the reference-pose run was abolished so that it is measured anew in each round, and the results document states ‘results can differ even when retested in the same pose’ and also shows the jam-search count for each trial.

Generalization. The lesson of Case D11 — replicas are not samples — holds when runs are replicas. Even in simulation, that re-runs with seed, configuration and code pinned produce the same result is not an assumption but a fact to be measured. When fixed-pose re-runs under the same conditions were gathered again, the groups whose trajectories were identical point for point were 8 of 39. The practice of reusing one run in a baseline hides the fact that the one run can be the outcome of a draw. The sensitivity of a test cell whose results split is not something to eliminate but something to measure and disclose (**PAVS-REP-06**), and the split ratio is itself the information of that test cell.

E. Reporting — how results are described

Case E1. Two score reports for the same test — with opposite results

Related clauses: **PAVS-REP-02** (attribution · phantom standards), **PAVS-VDT-02** (positive evidence), **PAVS-VDT-06** (hierarchy of authority), **PAVS-REP-03** (coverage list), **PAVS-REP-04** (no circularity of self-declaration)

Time of incident: 2026-08

Situation. Two paths issued score reports for the same test run: one a fail, the other "pass — certified for operation" (the indication at the time).

What went wrong, silently. The latter's pass was not made by the test data — **generating it from empty input produced exactly the same score report.** Absence was substituted with 0, default values were generated, and the requester specified the standard name as a parameter. As a separate finding in the same audit, yet another issuing path was entering on every score report a standard name not registered anywhere in the issuing system — the measurements required by the standard of that name had never been performed.

What caught it. The phantom-standard gate (refusal to issue under an unregistered standard name — firing confirmed in an actual issuing attempt) and attribution disclosure. After the fix, the score report stated on its own, item by item, "1 of authoritative origin, 12 self-defined", and a document that under the former format would have been shown as

‘passed the standard’ stated its own limits. (As of this edition, the score report issuing function is switched off.)

Generalization. A standard for results reporting is precisely **making the format itself state the grade of its own evidence**. A standard name is not a label chosen by the requester but a registered and verified attribution.

Case E2. The scoring was correct, but the results document described the opposite result

Related clauses: **PAVS-REP-13** (direct link to the verdict of record)

Time of incident: 2026-08

Situation. The scoring system recorded a pass for the track concerned, but the automatically generated results document (PDF) described the same test as a critical fail against an unrelated criterion.

What went wrong, silently. The document generator was not opening the verdict result file — through a chain of filename mismatches, the gap was filled by the threshold table of an unrelated domain. Even if the scoring is correct, if the display layer generates a parallel verdict, the final reader reads the wrong result.

What caught it. The results document was rewired to be generated directly from the verdict of record (no intermediate reinterpretation), and the document was made to carry its own ‘measurement reliability’ section — measured items, not-measured items and values that call for reservation.

Generalization. No reinterpretation layer is placed between verdict and reporting. A report shall be a rendering of the verdict of record and shall disclose its own reliability.

Case E3. Deleting half of the data still produced a ‘collection rate of 100%’

Related clauses: **PAVS-REP-06** (verification of statistics and metrics), **PAVS-VRF-01** (control-group principle)

Time of incident: 2026-07

Situation. A newly implemented data-collection health metric reported 100% under every condition.

What went wrong, silently. The denominator of the metric was "what was collected" — even with half deleted, the proportion collected among what was collected is 100%. If a metric refers to itself, no loss whatever is detected.

What caught it. A negative control applied to the metric itself: a known defect (deliberate deletion of data) was injected, and the metric was made to prove by a self-test (selftest) that it detects it — this control test detected the false metric at the pre-release stage.

Generalization. A metric carried in a results document shall itself prove that it can detect defects. A metric whose denominator is itself is not a metric.

Case E4. The accuracy "improvement" was a collapse reversed

Related clauses: **PAVS-REP-06** (null control · collapse metric)

Time of incident: 2026-08

Situation. A result came out that fine-tuning a verdict-support model had raised accuracy — on a balanced evaluation set (40 clips · 20 pass · 20 fail), from 20/40 (50.0% · Wilson 95% 35.2–64.8%) to 25/40 (62.5% · Wilson 95% 47.0–75.8%); the two intervals overlap (2026-08).

What went wrong, silently. The model before fine-tuning answered everything with one class (predicted fail 40 · pass 0), and the model after fine-tuning almost everything with the opposite class (predicted pass 35 · fail 5) — since the evaluation set was balanced, **merely flipping the direction of a single-class collapse arithmetically yields that margin of improvement.** The loss curve looked normal. In a separate improvement claim, an improvement margin that had been claimed without a control did not hold statistically and was withdrawn, and only when a null control passing through the same processing path alone was set up did the attribution of the improvement hold significantly — that control also rebutted the counter-argument that it was 'the effect of the processing path itself'.

What caught it. A ban on citing accuracy alone — the obligation to report the null control and the collapse metric (class distribution) alongside it.

Generalization. An improvement claim holds only when the null control and the collapse metric are reported alongside it. That a loss went down and that something was learned are different propositions.

Closing remarks (informative)

A single structure recurs in these cases: **unverified assumptions — that the value is a measurement, that the item is actually measuring, that the guard operates, that the results document reflects the verdict — silently collapsing.** Each clause of Part 1 turns one of those assumptions into an explicit obligation of verification.

The growing number of cases means that this specification is being maintained and revised. A new incident is a candidate for a new clause and is added to a revised edition of the Casebook.

PAVS Part 2 Casebook, draft v1.3 · 2026-10-20 · 39 cases (items 7 · measurement 7 · verdict 7 · verification 14 · reporting 4). The original records and evidence of each case are held by the publisher and are provided to partners in joint pilot testing.