Research Article

An Ungraded Survey Can Gate a 32.8% Assignment: Template Dates, Completion Restrictions, and Audit Blind Spots in Seven Moodle Course Instances

Julian Andrés Castaño Espinosa[1]

1. Corporación Unificada Nacional de Educación Superior, Bogotá, Colombia

Institutional learning management systems are rarely configured from scratch: courses arrive pre-populated from a template, and the instructor adjusts what the template got wrong. This report documents what an instrument-based audit of seven Moodle 4.5 course instances found when the adjustment was automated, and what the automation itself failed to see. The graded catalogue was clean: 53 graded items across the seven instances, all aligned to the departmental schedule, zero discrepancies. The defect lived entirely outside that catalogue. The 28 institutional feedback surveys — ungraded, and therefore absent both from the schedule table and from the client's activity-type map — kept their template dates: 13 of them had opening dates in 2028 and 2030 in a course running in 2026, and 11 of those 13 stored an opening instant identical to their closing instant, a window that can never open. We verify against the upstream Moodle 4.5 source that such a window passes form validation, because the check rejects only a close strictly earlier than the open. The consequential finding is a coupling: in two instances, the final assignment — weighted 32.8% of the course grade, with no grace period, since the cut-off date equals the due date — was restricted by *completion* of one of those surveys. The survey's opening date therefore was the assignment's effective opening date, leaving 2 usable days out of 31 in one instance and 10 out of 39 in the other, while the calendar displayed the full window. We report the correction (removing the survey windows rather than the institutional restriction), the safeguards that the incident forced into the writing path of the client, and the verification that nothing else moved: 206 existing survey responses intact, 0 of 19 writes altering a stored close date, and a 62-field before/after form diff showing exactly the 5 fields intended. We also report a limit that no instructor-level tool can pass: the deployment's survey form exposes no close-date field, although the upstream 4.5 source does, so 11 stored close dates remain in 2028 and 2030.

 doi.org/10.32388/Z2UXXT

The generalisable result is a rule for auditing template-populated courses: an item's grade weight predicts neither its schedule risk nor its blocking power, so any date audit must enumerate every activity type and resolve every access restriction before declaring a course correct. No student-level data are reported and no learning outcome is claimed.

Corresponding author: Julian Andrés Castaño Espinosa, julian_castanoe@cun.edu.co

1. Introduction

A course in an institutional learning management system (LMS) is seldom built by the person who teaches it. It is instantiated from a template that carries activities, resources, grade categories, access restrictions and — the point of this report — dates. The instructor then corrects what does not apply. The correction is normally done by hand, activity by activity, and is normally believed to be complete when the graded items look right, because the graded items are what students ask about and what the institution audits.

This report is about the part of the course that nobody audits because it carries no weight, and about the discovery that weight is a poor predictor of consequence. In the deployment studied, an ungraded institutional survey turned out to hold the key to a graded assignment worth 32.8% of the final grade, through an access restriction of the kind the platform calls conditional availability[1]. The survey's own opening date had been left at a template value in a future year. The assignment displayed its full submission window in the calendar and could not be submitted for all but the last two days of it.

Three things make this worth reporting rather than merely fixing. First, the failure is silent on both sides: the platform reports no error, the instructor's audit reports no discrepancy, and the student sees an open assignment that will not accept work. Second, the failure was invisible to the very instrument built to prevent it, and for a principled reason — the instrument enumerated the items that carry grades, which is exactly the criterion that excludes the offending activity.

Third, the underlying pattern is not specific to this institution: it is the general risk of a system in which the ordering constraints between activities (restrictions) are stored separately from the scheduling of those activities (dates), and in which the maintenance tooling is organised around grade weight.

We report what was measured, what was corrected, what could not be corrected at instructor level, and what the episode implies for anyone auditing a template-populated LMS course. We claim no effect on

student learning, report no student-level data, and make no counterfactual estimate of how many submissions the defect prevented — that number was not measured and is not inferable from the records available.

2. Setting and Instrumentation

2.1. The seven course instances

The audit covers the seven course instances taught by the author in the 2026-2 term at a private Colombian higher-education institution, on an institutional Moodle deployment identified internally as the institutional Moodle and recorded in the maintenance client as Moodle 4.5 — the open-source platform introduced by Dougiamas and Taylor[2], whose unit of organisation is the course as a container of activities, resources and the restrictions between them. The seven instances correspond to five distinct subjects — one subject contributes three parallel groups — and to two levels, undergraduate engineering and a specialisation programme. Aggregate enrolment across the seven instances at the time of the gradebook reading was 279 students. No student-level attribute of any kind was read, stored or reported in this work; the identifiers that appear below are Moodle course ids and course-module ids (cmid), which name platform objects, not people.

All seven instances were created from institutional templates. The template supplies the structure of the gradebook, the institutional survey activities, a large body of filler content, and the access restrictions between them.

2.2. The client and the verification protocol

Maintenance was performed with an authenticated HTTP client written for this purpose, which drives the ordinary teacher-facing web forms rather than a privileged interface: it reads /course/modedit.php?update=<cmid>, extracts every form field in document order, changes only the fields it intends to change, and posts the form back. The client holds no capability that the teacher role does not have, which matters for the interpretation of Section 3.6.

Every write follows the same protocol, and the protocol is the reason the findings below can be stated as measurements rather than impressions:

1. **Null round-trip before writing.** The unmodified form is posted back and the result compared, so that any field the client mangles is discovered before a real change is attempted.

2. **Minimal diff.** Only the fields corresponding to the requested change are altered; the complete before/after field list is compared and reported.
3. **Re-read from the server after writing.** Confirmation is taken from a fresh GET of the form and of the student-facing view, never from the response to the POST.
4. **Guard on collateral state.** For the surveys, the stored close date shown in the activity view — a value the form cannot edit, see Section 3.6 — is read before and after every write and reported if it changes.

Two independent instruments were used, and their agreement is part of the evidence. The maintenance client reads activity forms. A separate gradebook reader parses /grade/edit/tree/index.php and /grade/report/grader/index.php to recover effective weights, scales, aggregation methods and per-item counts of graded submissions. The 32.8% weight quoted throughout was read by the second instrument, not asserted by the first.

2.3. What counts as measured here

Every figure in Section 3 comes from one of three sources: a form read from the deployment, a gradebook page read from the deployment, or the upstream Moodle source code retrieved from the project's public repository on the branch matching the deployed version. Where a quantity was not measured, this report says so and does not estimate it. In particular, no claim is made about the number of students who attempted a submission and were blocked, about grade impact, or about the frequency of this configuration in other institutions.

A note on identifiers. *Moodle course and module identifiers have been replaced with neutral labels (Instance A, B, ...). They are internal to the deployment and carry no meaning outside it; the substitution is one-to-one and consistent throughout, so every claim remains traceable within the paper.*

3. Results

3.1. The graded catalogue was consistent

The first pass adjusted every graded item in the seven instances against the departmental schedule held in the repository. The result was clean, and this is the baseline against which the rest of the report should be read.

Metric	Value
Course instances	7
Graded items adjusted	53
Quizzes	38
Assignments	8
Graded forums	7
Items with a discrepancy after adjustment	0
Items skipped	0

The gradebook reader independently confirmed, for each instance, that every catalogue item held in the repository was present in the gradebook, that effective weights matched the repository and summed to 100.0%, that every grade category aggregated as a weighted mean, and that the graded items shared a single 0.00–5.00 scale. In one instance it also confirmed that no gradebook item lay outside the catalogue; in the other six it did not, and what it found there instead is reported in Section 3.7.

Under any audit organised around assessment weight, the seven courses were correct.

3.2. The 28 ungraded surveys were never inspected

They were not correct. Each instance carries institutional feedback activities — three teaching evaluations and one learning-environment survey — 28 in total across the seven instances. None of them was touched by the first pass, for two compounding reasons that are worth separating because they generalise differently.

The first reason is that the date command never walks the course: it walks the schedule. For each item in the repository's schedule table it looks up an activity of the same name inside the course and writes that activity's dates. The schedule table is indexed by assessment item, because it is the assessment schedule, and ungraded activities are not in it — so the 28 surveys were never candidates for the writer, and the command's closing summary, which counts the items it was handed, reported nothing about them. What

the tool does not enumerate, the tool cannot fix, and — worse — the tool's clean report says nothing about it.

The second reason would have blocked the fix even if the first had not. The client's date-field map associated a set of form field names with each activity type, and it covered three: quiz (timeopen, timeclose), assign (allowsubmissionsfromdate, duedate, cutoffdate, gradingduedate) and forum (duedate, cutoffdate). The type feedback was absent, so a survey that did reach the writer would have been declined for want of a field map; and even with feedback added, the command would still have had no date to write for it. The correction therefore required a separate sub-command with its own policy, not a new branch of the existing one.

3.3. Thirteen surveys that could not open

A census of all 28 surveys, reading timeopen from each activity's edit form and the displayed window from each activity's view page, found 13 with a stored opening date in the template years 2028 and 2030 — in a course term running from August to November 2026.

Course instance	Surveys with template opening	Stored opening values
Creatividad y Pensamiento Innovador (Instance B)	4	27 Oct 2030
Investigación en Ciencia y Tecnología (Instance A)	4	18 Feb 2028
Trabajo de Grado 2 (129268)	1	18 Feb 2028
Trabajo de Grado 3, group 54450 (112321)	1	18 Feb 2028
Trabajo de Grado 3, group 54466 (116387)	1	18 Feb 2028
Trabajo de Grado 3, group 54467 (129270)	1	18 Feb 2028
Proyecto I, specialisation (130378)	1	18 Feb 2028
Total	**13**	

Of the 13, **11 stored an opening instant identical to the closing instant**, to the minute: three at 27 Oct 2030 05:31, one at 27 Oct 2030 05:35, five at 18 Feb 2028 11:04 and two at 18 Feb 2028 11:05.

That is a window of zero duration — an activity that opens and closes at the same instant and is therefore never available. The remaining 2 had a future opening and no stored close at all.

The other 15 surveys had the opening checkbox disabled altogether. They displayed no window, were permanently available, and contained 206 real student responses. They were functioning precisely because they had no dates.

The symptom that started the investigation was a student report: an attempt to answer the learning-environment survey returned a generic error message. The teaching-evaluation activity of one instance displayed, to the instructor, an opening and a closing at the same minute in February 2028.

3.4. Why a zero-duration window is legal

A window whose close equals its open looks like something a form should reject. It is not rejected, and the reason is visible in the upstream source of the version deployed here. In mod/feedback/mod_form.php on branch MOODLE_405_STABLE, the validation method contains a single date consistency check, which raises an error only when the close is *strictly earlier* than the open[3]. Equality passes. A template that stamps the same timestamp into both fields — which is what the observed values look like, since each pair carries one and the same timestamp and those timestamps take only four distinct values across the seven instances — produces a valid form submission and a permanently unavailable activity.

This has an operational consequence for anyone writing an audit rule, and it is the reason the classification order in Section 4.3 matters: a zero-duration window in the *past* is indistinguishable, to a naive check, from an activity that opened normally and is now simply available. The naive rule "flag activities whose opening lies in the future" would have reported these 13 today and reported nothing at all from 19 February 2028 onwards, at which point they would be permanently broken and permanently invisible to the audit.

3.5. The coupling: an ungraded survey gating a 32.8% assignment

The first corrective pass gave the 13 broken surveys ordinary 2026 opening dates aligned to the grading periods. Re-reading the courses after that pass surfaced the finding that changed the decision, and that is the substantive contribution of this report.

In two of the seven instances, the final assignment carries an access restriction of type *completion* pointing at the third teaching-evaluation survey: literally {"type":"completion","cm":<cmid of the survey>,"e":1} in the module's stored availability conditions, with the module reporting

hascmrestrictions. The survey, in turn, is configured to count as complete only when it is *answered* (completionview together with completionsubmit), and it cannot be answered before it opens.

The consequence is a silent transfer of scheduling authority. The assignment's own dates were correct. Its effective opening date was the survey's opening date.

Course instance	Final assignment	Its own window	Gating survey	Usable days before	Usable days after
Investigación (Instance A)	cmid 6522210	opens 13 Aug, cut-off 12 Sep 23:59	cmid 6522208, opened 11 Sep	**2 of 31**	31 of 31
Creatividad (Instance B)	cmid 6745731	opens 12 Aug, cut-off 19 Sep 23:59	cmid 6745729, opened 10 Sep	**10 of 39**	39 of 39

Usable days are counted inclusively between the effective unlock and the cut-off. Two aggravating properties of these assignments matter. First, the weight: the gradebook reader records the final assignment at **32.8%** of the course grade in both instances — the largest single item in the undergraduate scheme, ahead of the first partial examination at 24.0%. Second, the absence of any grace period: the cut-off date equals the due date, so the platform accepts nothing after the deadline, late or otherwise. A student who discovered the assignment on the last available day had no recourse inside the platform.

Nothing in the student's view announces the coupling as a scheduling fact. The calendar shows the assignment's window. The restriction is displayed as a condition to satisfy, not as a date, and the condition is on an activity that carries no grade and is presented as an institutional formality.

3.6. A field the deployment's form does not expose

Correcting the openings was possible. Correcting the closes was not, and the reason is a difference between the deployed forms and the upstream code.

Measured across all 28 surveys: in /course/modedit.php?update=<cmid> for a feedback activity in this deployment, the only date selectors present are timeopen and completionexpected. The string timeclose does not occur anywhere in the served HTML — not as a field, not as a label. The fieldset legends read

"Availability" and "Allow answers from". The close date that the student-facing view does display comes from the stored record and cannot be reached through the form.

The upstream source for the same version does define that field: mod/feedback/mod_form.php on MOODLE_405_STABLE adds a date_time_selector for timeclose immediately after the one for timeopen[3]. The deployed form therefore diverges from the upstream form of the version the client identifies. This report does not establish the cause of the divergence — a local customisation, a plugin, a role-level form restriction and a version mismatch are all consistent with what was observed, and distinguishing them requires access this author does not have.

What is established is the operational consequence, and it is not small: **11 stored close dates remain in 2028 and 2030**, and they are the only dates in the seven instances that are not 2026 dates. The audit counted 149 active dates in all — 135 on activity forms and 14 at course level — so 138 of them are 2026 dates and 11 are not. Those 11 belong to institutional teaching-evaluation surveys, which will therefore keep accepting responses between roughly seventeen months and four years after their courses close grades. Two further surveys have no stored close at all and remain open indefinitely; they were not emptied of a close date, they never had one.

Course-level dates are equally out of reach: they arrive frozen for the teacher role, and only an administrator can move them.

3.7. The same blind spot on a second axis

The pattern found in the date audit — that the graded catalogue is well tended and everything outside it is not — was independently visible in a different maintenance task performed on the same courses, and the agreement is worth recording because it suggests the blind spot is structural rather than incidental.

A visibility pass classified every course component by whether it carries weight in the gradebook, reading the weight from the gradebook tree rather than inferring it from the component's name. Across the seven instances the pass left **168 components visible and hid 101, with 0 errors**, every action reversible by a single command. All but one of the hidden components were template filler: SCORM packages and pages named "Contenido N (haz clic aquí)", placeholder links named "G1 … G8", "Video 1 … Video 8" and "Recurso 1 … Recurso 8" — one set per topic — interactive-image and podcast pages, an institutional "Modelo pedagógico" page in every instance, and empty text-and-media areas.

The single exception was a test forum the author had created himself. The instance with the largest amount of filler contributed 35 hidden components; three instances contributed 2 each.

The case that prompted that pass is a useful miniature of the whole problem. A pair of template forums, visible to students, had an empty description and a single existing discussion whose opening post had an empty body, stamped by the template. Both were of the "single simple discussion" type, in which a student cannot open a new thread and can only reply to the existing — blank — message.

Neither had a gradebook item at all and neither was rated, so neither could produce a grade. The same pair was then found, also empty and also visible, in a second instance. Students asked what they were supposed to do there, which is the only reason anyone looked.

The gradebook reader, working from a different page, corroborated the same phenomenon from the other direction: it counted **18 template filler items that do have a gradebook entry but carry 0% weight**, distributed across six of the seven instances (8, 1, 1, 1, 1 and 6), on 0.00–1.00 and 0.00–100.00 scales, and verified that they move no grade. Two instruments, two pages, one conclusion: in a template-populated course, the set of components that matter and the set of components that look official are different sets, and the difference is not legible from a component's name.

4. The Correction and Its Safeguards

4.1. What was changed, and what deliberately was not

The corrective decision reversed the first pass. Rather than schedule the teaching-evaluation surveys sensibly, all six that carried windows were left with **no opening date at all** — the configuration that the other 15 surveys already had, and the configuration under which they work. Six writes, each verified by re-reading the server. The learning-environment survey in all seven instances received a single opening date, 10 August 2026, the term start date recorded for the five subjects; it is in the past, so the activity is available now, and it is the only survey date the repository actually declares.

Two things were deliberately not changed. The access restriction on the final assignment was left exactly as the template placed it, still pointing at the survey's completion, verified unchanged by re-reading the server: it is an institutional requirement that students evaluate their teaching, and removing it would silently remove that requirement. Once the survey is always available, the lock can be opened on any day of the assignment's window, so the restriction stops being a scheduling constraint and becomes what it

was meant to be — a task ordering. The 15 already-working surveys were not touched either, for the reason given in Section 4.2.

The other access restrictions in these two instances were checked and were consistent: the first quiz and first partial with the first evaluation, the second pair with the second, the third quiz with the third, and the self- and peer-assessment activities with the learning-environment survey, which opens at term start.

4.2. The guard belongs on the writing path

The 15 functioning surveys had their opening checkbox disabled and contained 206 student responses.

Giving one of them a window would have closed a live institutional data collection. The census sub-command already refused to touch them, but the census is the component that *decides*; the component that *writes* is a generic date setter shared with quizzes and assignments, and it emits an enabled flag for every date field it sets. Adding feedback to the type map therefore re-opened, at the level of the shared writer, exactly the risk the census had closed at the level of the policy.

The guard was moved to the writer: if a survey arrives with its opening disabled, the writer does not enable it and says so. The regression check is a dry-run plan over the two instances involved in the incident, which still proposes the same 8 graded items each, zero surveys, zero problems.

The general form of this lesson: a safety rule enforced in the component that chooses actions is not enforced at all once a second caller reaches the component that performs them.

4.3. Ordering the classification rules

The census classifies each survey into one of four states, and the order in which the tests are applied is load-bearing:

State of the opening	Verdict
Disabled	Working, do not touch — never enable a disabled window
Enabled, and open equals close	Zero-duration window: broken, correct it
Enabled and already past	Already opened, do not touch
Enabled and still in the future	Never opens: broken, correct it

The zero-duration test is evaluated *before* the already-past test. With only the other three rules, the defect that motivated all of this work — open equals close equals 18 February 2028 11:04 — would have been reported as broken until that date and then reclassified as “already opened, do not touch” from 19 February 2028 onwards, while remaining permanently closed. A rule set can lose a defect by the passage of time.

A companion correction was required in the parser that reads the displayed window: Moodle writes past labels in a different grammatical form than future ones, and the parser recognised only the future form. This mattered twice over. It made every survey corrected to a past date print an empty window, indistinguishable from having lost its opening; and, more seriously, it would have blinded the only guard watching a field the form cannot edit, since a stored close moving into the past would be labelled in the past form and read as absent both before and after a write.

4.4. Evidence that nothing else moved

The final census re-read all 28 surveys from the server after both passes.

Check	Result
Surveys with an opening in 2028 or 2030	0 (were 13)
Surveys with a zero-duration window	0 (were 11)
Permanently available, no window	21 (the 15 that already worked, plus the 6 rectified)
Learning-environment surveys with an opening	7, at 10 Aug 2026 00:00, already open
Student responses in the untouched group	206 of 206, counted survey by survey
Untouched surveys that acquired a window	0
Surveys left hidden by the operation	0 of 28
Writes that moved a stored close date	0 of 19
Final-assignment restriction still in place	Yes, in both instances, and satisfiable from day one

The strongest single piece of evidence that the writes were surgical is the field-level diff: for each survey

written, the complete before/after comparison of the 62 form fields shows **exactly 5 differences**, the five components of the opening date-time selector, with the enabled flag set both before and after. Visibility, description, question count, item identifiers and responses are identical. The stored close dates retain their original template timestamps to the minute — 11:04, 11:05, 05:31, 05:35 — which is itself the proof that no write reached them.

5. Discussion

5.1. Three compounding causes

The defect reported in Section 3.5 required three independent conditions, none of which is a programming error in the platform.

A template carrying dates from another instance. Template dates in 2028 and 2030 are not plausible values for any current course; they are residue from whatever instance the template was captured in. The platform accepts them because there is nothing incoherent about a course scheduled in 2030, and accepts the zero-duration form of them because the validation rejects only strict inversion (Section 3.4).

Maintenance tooling organised by grade weight. Both the schedule table and the client's type map enumerate what is assessed. This is a reasonable organising principle — it is the principle under which the 53 graded items came out correct — and it is precisely the principle that excludes the activities that turned out to hold the keys. Weight predicts audit coverage; it does not predict blocking power.

Restrictions stored separately from schedules. The platform models "when is this available" and "what must be done first" as different mechanisms, and correctly so: they answer different questions[1]. But when the prerequisite is itself schedulable, the composition of the two mechanisms yields an effective schedule that neither mechanism displays. This is a configuration composition hazard rather than a fault, of the family documented in production systems generally[4][5], aggravated by the sheer number of independently settable options a modern LMS activity exposes[6], and it has the property those authors emphasise: the individual settings are each defensible, and the failure exists only in their interaction. That is the standard anatomy of an accident in a tightly coupled system — a latent condition planted by one decision and triggered by another, with no single actor in a position to see both[7][8] — and it argues, as Leveson[9] does for safety generally, for treating the constraint between components, rather than each component's own correctness, as the thing to be enforced. It is also, in the vocabulary of technical debt, a

form of configuration debt that accrues at instantiation time rather than at development time[10][11][12], and that the person who inherits the course pays.

5.2. A rule practitioners can apply

The audit rule that follows from this episode is short enough to state as a sentence and can be carried out with the ordinary teacher interface: **before giving an opening date to any activity, determine which items name it as a completion condition, and confirm that none of them opens earlier. **Its converse is equally useful:** when an assignment's usable window is shorter than its displayed window, look for a completion condition pointing at an ungraded activity.**

Three corollaries earned in this work are worth stating with it. An activity whose window cannot be edited from the form is a permanent liability, so its state must be watched rather than assumed. A safety rule must live in the code that writes, not only in the code that decides. And a rule set that classifies by date must test for degenerate windows before testing for "already past", or it will lose the defect exactly when it becomes unfixable.

5.3. Why the visible calendar is not the effective calendar

The student-facing consequence is the part with pedagogical weight, and it is worth naming precisely, because it is not a usability nuisance. The platform displayed an assignment window of 31 days and enforced one of 2. A student planning against the displayed window — which is the only window available to them — plans against a fiction. The literature on deadlines and self-regulation is built on the assumption that the stated deadline is the operative one[13][14], and the design literature on assessment assumes that students can see the conditions under which they are assessed[15][16][17]. A hidden gate breaks that assumption without announcing itself, and it does so asymmetrically by construction rather than by anything measured here: it is the student who starts early who meets the block, with no way to tell from the interface whether it is temporary. The mismatch between displayed state and enforced state is the classic violation of visibility of system status[18][19], here with a consequence measured in grade weight rather than in frustration.

There is also a compliance cost of the kind Herd and Moynihan[20] call a learning cost: the student must work out, unaided, that an ungraded formality is the thing standing between them and a submission. The visibility findings of Section 3.7 raise that cost further, since a course populated with 101 components

that turned out to be filler offers no reliable way to tell an institutional formality that matters from one that does not.

5.4. Threats to validity and limitations

Single institution, single term, single operator. All seven instances belong to one deployment, one academic term and one instructor. The specific numbers are not a sample of anything; they characterise this deployment. What generalises is the mechanism and the audit rule, not the prevalence.

No outcome was measured. We do not know how many students attempted the blocked assignment, and we make no claim about grades. The defect was corrected on 17 August 2026, before the first affected due date, so no final grade in these instances was computed under the blocked configuration. Any statement about learning impact would be unmeasured, and none is made; the traces an LMS keeps describe activity on a platform rather than learning, and reading them as the latter is precisely the inference this report declines to make[21].

The divergence in Section 3.6 is not explained. We establish that the deployed form lacks a field that the upstream source of the identified version defines, and we establish the consequence. We do not establish why, and we could not: diagnosing it requires administrative access.

The correction is verified, not proved permanent. Verification is a re-read of the server at a point in time. Templates are re-applied at each new term, and nothing in this work prevents the same dates from arriving again in 2027. What the work leaves behind is the census that will detect them.

Instrument bias. Both instruments were written by the author, and a systematic misreading of the gradebook tree or of the activity form would be invisible to both. The mitigations were independence of source page and cross-checking of the quantities both instruments can see — the presence of a graded item and the dates it carries, which the maintenance client writes on the activity form and the gradebook reader recovers from the gradebook — and these agreed for all 53 items.

6. Conclusions

An audit of seven Moodle 4.5 course instances found the graded catalogue in order — 53 items, zero discrepancies — and the defect entirely outside it. Of 28 ungraded institutional surveys, 13 carried template opening dates in 2028 and 2030, and 11 of those stored an opening instant identical to their

closing instant: a window that never opens, and that passes upstream form validation because the check rejects only a close strictly earlier than the open.

The consequential finding is the coupling. In two instances the final assignment, weighted 32.8% with no grace period, was gated by *completion* of one of those surveys, so the survey's opening date was the assignment's effective opening date: 2 usable days of 31 in one instance, 10 of 39 in the other, against a calendar that displayed the full window. The correction removed the surveys' windows rather than the institutional restriction, and was verified by re-reading the server: 206 existing responses intact, 0 of 19 writes touching a stored close date, and a 62-field diff showing exactly the 5 intended changes.

One limit could not be passed at instructor level: the deployment's survey form exposes no close-date field although the upstream source of the identified version defines one, leaving 11 stored close dates in 2028 and 2030 out of 149 active dates. That is reported here as an open defect requiring administrative action, not as a solved problem.

The generalisable result is a rule and a warning. The rule: before scheduling any activity, resolve which items name it as a completion condition; and when an assignment's usable window is shorter than its displayed window, look for a completion condition on an ungraded activity. The warning: maintenance tooling organised around grade weight will produce a clean report about a broken course, because in a template-populated LMS the components that carry no weight can still carry the keys.

Statements and Declarations

Funding

No external funding was received for this work.

Potential Competing Interests

The author declares no conflict of interest. The author is an instructor at the institution whose learning management system deployment is described, and audited his own courses; he holds no financial or commercial relationship with Moodle Pty Ltd., with the deployment's operator, or with any vendor named in this manuscript, and received no payment or benefit conditioned on the content of this report.

Ethics

This manuscript contains no personally identifying information about any student: no names, no identity or enrolment numbers, no individual grades, no submission contents and no survey responses. All student-related quantities reported are aggregates at the level of a course instance (enrolment counts, response counts, per-item counts of graded submissions). The identifiers that appear in the text are Moodle course and course-module identifiers, which designate platform objects. The work is an administrative audit of the author's own course configuration, carried out with the ordinary permissions of his teaching role; it involved no research with human subjects, no intervention applied to students, no collection of data from students, and no processing of personal data beyond the incidental exposure inherent in a teacher viewing his own gradebook. For that reason no ethics committee approval was sought, and none is required under the institution's rules for administrative work of this kind. The single student-originated observation quoted — an error message received when opening a survey — is reported without any attribute of the person who reported it.

Data Availability

No research dataset was generated by this work, and no dataset can be shared. The measurements reported are readings of an authenticated institutional learning management system that is not publicly accessible and whose pages contain enrolled students' personal data; publishing exports of those pages would be neither lawful nor possible. What is reported here are aggregate counts and platform object identifiers (course ids and course-module ids), which are included in the text so that the institution's own administrators can reproduce every reading. The two verifiable external sources are cited in full and are public: the upstream Moodle source file and branch used in Sections 3.4 and 3.6, and the platform's availability documentation. The audit procedure is described in Section 2.2 in sufficient detail to be reimplemented against any Moodle deployment by a reader with teacher-level access to their own courses.

Author Contributions

Conceptualisation, methodology, software, data curation, formal analysis, original draft, review and editing: the single signing author.

Use of Generative AI

Generative artificial intelligence assistants — Claude (Anthropic) and ChatGPT (OpenAI) — were used in the preparation of this manuscript for drafting and editing prose, for translating the abstract, and for checking internal consistency between the figures reported in the text and the audit records they come from. They were also used while writing the maintenance client described in Section 2.2. No assistant generated measurements, and no figure in this manuscript was produced by an assistant: every quantity was read from the deployment or from the cited source code, and was checked individually by the author. No bibliographic reference was accepted without individual verification.

The author accepts full responsibility for the content of the manuscript.

References

1. ^[a], ^[b] *Moodle Pty Ltd (n.d.). "Availability API (Version 4.5) [Developer Documentation]." Moodle Developer Resources. https://moodledev.io/docs/4.5/apis/subsystems/availability.*
2. ^ *Dougiamas M, Taylor P (2003). "Moodle: Using Learning Communities to Create an Open Source Course Management System." Association for the Advancement of Computing in Education. pp. 171–178.*
3. ^[a], ^[b] *Moodle Pty Ltd (2024). "Mod/feedback/mod_form.php (Branch MOODLE_405_STABLE) [Source Code]." Moodle. https://github.com/moodle/moodle/blob/MOODLE_405_STABLE/mod/feedback/mod_form.php.*
4. ^ *Yin Z, Ma X, Zheng J, Zhou Y, Bairavasundaram LN, Pasupathy S (2011). "An Empirical Study on Configuration Errors in Commercial and Open Source Systems." Association for Computing Machinery. pp. 159–172.*
5. ^ *Xu T, Zhou Y (2015). "Systems Approaches to Tackling Configuration Errors: A Survey." ACM Comput Surv. **47**(4):Article 70.*
6. ^ *Xu T, Jin L, Fan X, Zhou Y, Pasupathy S, Talwadker R (2015). "Hey, You Have Given Me Too Many Knobs! Understanding and Dealing with Over-Designed Configuration in System Software." Association for Computing Machinery. pp. 307–319.*
7. ^ *Perrow C (1984). Normal Accidents: Living with High-Risk Technologies. Basic Books.*
8. ^ *Reason J (1990). Human Error. Cambridge University Press.*
9. ^ *Leveson NG (2011). Engineering a Safer World: Systems Thinking Applied to Safety. MIT Press.*
10. ^ *Cunningham W (1992). "The WyCash Portfolio Management System." OOPS Messenger. **4**(2):29–30.*
11. ^ *Kruchten P, Nord RL, Ozkaya I (2012). "Technical Debt: From Metaphor to Theory and Practice." IEEE Softw. **29**(6):18–21.*

12. ^*Sculley D, Holt G, Golovin D, Davydov E, Phillips T, Ebner D, Chaudhary V, Young M, Crespo JF, Dennison D (2015). "Hidden Technical Debt in Machine Learning Systems." pp. 2503–2511.*
13. ^*Ariely D, Wertenbroch K (2002). "Procrastination, Deadlines, and Performance: Self-Control by Precommitment." Psychol Sci. **13**(3):219–224.*
14. ^*Steel P (2007). "The Nature of Procrastination: A Meta-Analytic and Theoretical Review of Quintessential Self-Regulatory Failure." Psychol Bull. **133**(1):65–94.*
15. ^*Gibbs G, Simpson C (2005). "Conditions Under Which Assessment Supports Students' Learning." Learn Teach High Educ. **1**:3–31.*
16. ^*Nicol DJ, Macfarlane-Dick D (2006). "Formative Assessment and Self-Regulated Learning: A Model and Seven Principles of Good Feedback Practice." Stud High Educ. **31**(2):199–218.*
17. ^*Boud D, Molloy E (2013). "Rethinking Models of Feedback for Learning: The Challenge of Design." Assess Eval High Educ. **38**(6):698–712.*
18. ^*Nielsen J (1994). "Enhancing the Explanatory Power of Usability Heuristics." Association for Computing Machinery. pp. 152–158.*
19. ^*Norman DA (2013). The Design of Everyday Things. Rev. and expanded ed. Basic Books.*
20. ^*Herd P, Moynihan DP (2018). Administrative Burden: Policymaking by Other Means. Russell Sage Foundation.*
21. ^*Gašević D, Dawson S, Siemens G (2015). "Let's Not Forget: Learning Analytics Are About Learning." TechTrends. **59**(1):64–71.*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.