Nobody Handles the Error: Reliability Engineering in Published Automation Templates

A structural analysis of 5,147 n8n workflow templates

Abstract

Automation platforms let people wire third-party services together without writing code, and their public template galleries are a written record of how that work is assembled. We analyse the reliability configuration of 5,250 deduplicated n8n templates from the largest AI workflow marketplace Neura Market https://www.neura.market/workflows — every template we could parse from a corpus of 13,126 — by reading each template's own executable specification rather than its description. 5,147 of them call at least one external system, and those are the subject of every figure below.

The result is one-sided. 81.0% (95% CI 79.9-82.1) of templates that call an external system declare no error handling of any kind: no node-level error policy, no followed error branch, no workflow error handler, no error trigger. Counting individual calls rather than templates, 5.2% of 40,863 external calls declare any error handling, 2.44% declare a retry, and 0.33% declare a timeout. The median template makes 6 external calls and protects none of them.

Two secondary results. Integrations co-occur in 6 distinct communities recoverable from structure alone. And AI-containing templates carry fewer deterministic branching steps than templates matched on size — a small effect (Cliff's delta -0.224) in the direction that displacement of deterministic logic by model calls would predict.

Ten candidate anti-pattern detectors were manually audited; the audit rejected three of them outright and each rejection was an implementation that did not match its own definition.

Published templates are artefacts of intent, not evidence of deployment. What they show is that the reliability engineering an automation needs in production is almost entirely absent from the examples people copy.

1. Introduction

A workflow automation is a small distributed system. It calls services it does not control, over networks that fail, on behalf of a person who is not watching. The engineering that makes such a

system survive contact with production is well understood and unglamorous: retry the call that failed, bound how long you wait, make the write safe to repeat, and send the failure somewhere a human will read it.

Public template galleries are where people go to learn how to build these. n8n, Make, Zapier and others publish thousands of templates that can be imported and run. They function as worked examples, and worked examples teach.

This report asks what those examples declare about failure.

These are templates, not deployments — and that shapes everything that follows. The corpus was collected from n8n's public template gallery. We cannot observe whether any template here was ever imported, let alone run. A template that omits a retry may sit beside a production workflow, in the same author's account, that has one. Everything measured here is *declared configuration in a published artefact*, and the report says "templates" rather than "automations" throughout for that reason.

That limitation is real, and it is also narrower than it first appears. A published template is a claim about how a job should be done. It is copied, adapted and shipped. If the examples people copy contain no error handling, the omission propagates whether or not the original author knew better.

We make three contributions. First, a reliability measurement across 5,250 templates covering six mechanisms, reported under three explicitly stated denominators. Second, a set of deterministic detectors for structural anti-patterns, with their prevalence and their observability limits stated. Third, the code and the per-statistic outputs, so that any number here can be recomputed or contradicted.

This report follows NM-TR-2026-01, *What Builders Publish, and What Gets Used*, which analysed the same corpus for structure and adoption. That report measured error handling with a single boolean and did not examine retries, timeouts, idempotency or recovery paths. This one does. Where a finding of that report is needed as context it is cited, not restated.

2. Related work

Workflow corpora. Prior structural analysis of no-code automation has been mostly n8n-only and mostly descriptive: node counts, popular integrations, growth of AI nodes. NM-TR-2026-01 extends this to two platforms and relates structure to each platform's own adoption counter. Neither that work nor its predecessors examine failure handling beyond presence or absence.

Reliability in distributed systems. The mechanisms measured here — bounded retry with backoff, timeouts, idempotency keys, dead-letter queues — are standard practice in message-driven architecture and are treated as table stakes in that literature. The gap this report addresses is that no one has asked how much of that practice survives into the low-code layer, where the

person assembling the system is frequently not a distributed-systems engineer and the platform defaults decide what happens.

Declared versus observed configuration. Studies of infrastructure-as-code and CI configuration face the same limitation this one does: a file records intent, not behaviour. The convention in that literature is to state the gap plainly and measure the artefact anyway, because the artefact is what gets copied. We follow it.

---

3. Methods

3.1 Preregistration

The analysis plan, inclusion criteria, feature definitions, denominators and seven conditional decision rules were written to `preregistration.md` and committed before any script read the corpus. Three of those rules fired and are reported where they apply (R3, R4, R7). One did not and is reported as not taken (R1). Nothing in this report was chosen after seeing its result.

3.2 Corpus

The corpus is the frozen snapshot `neura-automation-study-2026-08-17`, read-only, shared with NM-TR-2026-01 so that both reports describe the same templates. Of 13,126 n8n rows, 12,892 parse into a step graph.

Fields known to be synthetic in the source database — ratings, prices, `featured`, author identities, n8n download counts and `created_at` — are not read by any script in this study. The evidence is each template's own specification.

3.3 Why n8n only

Make's scenario format exposes no per-module error field. Make error handling is **unobservable, not absent**, and pooling the two platforms would code "not recorded" as "not configured". Every reliability figure in this report is n8n-only, and Make appears in no reliability comparison. Zapier (334 templates), Activepieces (65) and Pipedream (23, none parseable) are reported as a coverage gap and excluded.

3.4 Deduplication

Marketplace corpora contain forks and reskins, and counting them separately inflates every frequency. We compute a canonical graph hash per template — the sorted step-type multiset plus edge shape, ignoring node names, positions, identifiers and credentials — and collapse exact collisions.

A second, near-duplicate stage using multiset Jaccard was implemented and then **removed**: manual adjudication measured its precision at 0.17 and no threshold repaired it. Section 4.7 reports what that inspection found. Deduplication in this report is the exact-hash stage alone.

Deduplication removes 59.3% of parseable templates, leaving 5,250 templates of which 5,147 make an external call. That exceeds the 25% threshold of decision rule R4, so the deduplicated figure leads throughout and the pre-deduplication figure is reported beside it.

The loss is not a hashing artefact. The largest cluster is 33 members of one 16-step agent scaffold — a chat trigger, a director agent, and six sub-agents — republished with the domain labels changed from LinkedIn content to social media to creative design. Templates of this kind are one artefact, not thirty-three observations.

Figure 1 reports the flow from retrieved templates to the analysed set, with the count dropped at each step.

Figure 1. Corpus flow

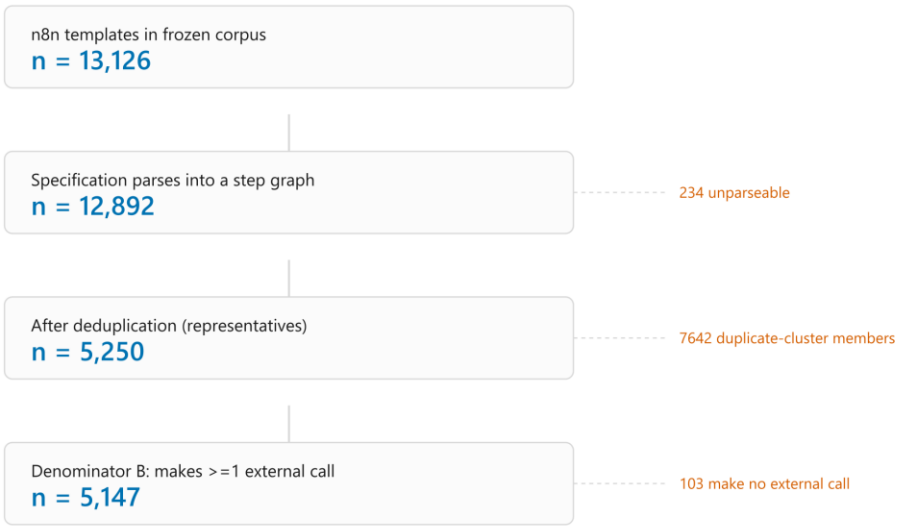


Figure 1. Flow from retrieved n8n templates to the analysed set. Deduplication is by canonical graph hash then near-duplicate clustering. Neura Market Workflow Corpus Study (2026). CC BY 4.0 - https://www.neura.market/research/workflow-corpus-study

Flow diagram. 13,126 n8n templates in the frozen corpus; 12,892 parse into a step graph; 5,250 remain after deduplication; 5,147 of those make at least one external network call and form denominator B. Neura Market Workflow Corpus Study (2026). CC BY 4.0.

3.5 What counts as an external call

A step makes an external call when it reaches a third-party system: an HTTP-family node, a raw transport, or a node bound to an identifiable application. Control flow, data shaping and documentation nodes do not, whatever verb they declare. n8n sticky notes are documentation and are excluded from every count, as are LangChain sub-nodes, which are attachments to an AI step rather than steps.

3.6 Mechanisms

Six mechanisms, each detected from declared configuration: node-level error policy (`onError`, `continueOnFail`, `retryOnFail`); a followed error branch; a workflow-level error handler; a node or workflow timeout; an idempotency guard (an explicit deduplication step, an upsert verb, or a read on an application immediately upstream of a write to it); and a recovery path — an error branch terminating in a store or notification, or performing an alternative action.

One trap is worth recording because it would invert the headline. n8n overloads output port 1. It is the error path only on a node that sets `onError = continueErrorOutput`; on an `if` it is the false branch, on a `switch` the second case, on `splitInBatches` the loop output. Counting port-1 edges as error branches inflates error handling by roughly an order of magnitude. Every detector here gates on the `onError` value before reading the port.

3.7 Denominators

Every proportion states its own denominator.

- **A** — all analysed templates, 13,126 after deduplication.
- **B** — templates making at least one external call, 5,250.

A pure data transform does not need a retry, so B is the fair denominator and the abstract leads with it.

- **C** — individual external-call steps, 40,863

pooled across templates. A template-level boolean overstates reliability when one call of eleven is configured; C is the step-level view.

3.8 An observability ceiling on idempotency

n8n does not serialise a defaulted `parameters.operation`, so a verb is visible on a median 11.1% of external-call steps (2333 templates declare none at all). Upsert and read-before-write detection is bounded by that. **Every idempotency figure in this report is a floor, not an estimate**, and is labelled as one.

3.9 Statistics

Proportions carry a 95% Wilson score interval, chosen over the normal approximation because several of these proportions sit near zero, where the normal interval produces a negative lower bound. Distributions are reported as median and interquartile range, not mean and standard deviation, because step and call counts are right-skewed. Effect sizes accompany every comparison; at these sample sizes significance is guaranteed and uninformative on its own.

Nothing here is causal. Templates are not randomly assigned their features, and platform, author, age and purpose are confounded with everything measured. The word used is "associated".

4. Results

4.1 Most templates that call external systems declare nothing about failure

81.0% (95% CI 79.9-82.1) of templates making at least one external call declare no error handling of any kind — no node-level error policy, no followed error branch, no workflow error handler, no error trigger. Denominator B, n = 5,147.

The single most common mechanism is a node-level error policy, present in 17.5% (95% CI 16.5-18.6) of denominator B (n = 5,147). No mechanism reaches one template in five.

Retry is declared by 7.5% (95% CI 6.8-8.3) of denominator B (n = 5,147).

A timeout — at either node or workflow level — is declared by 2.1% (95% CI 1.8-2.6) of denominator B (n = 5,147).

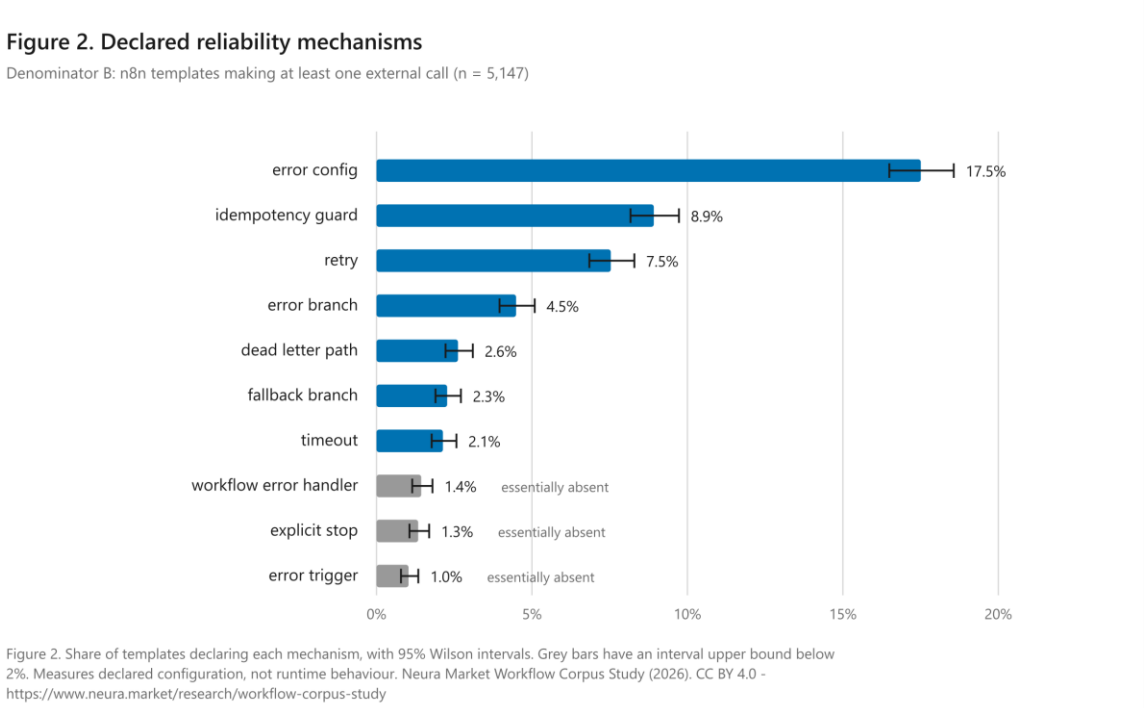
An error branch that is actually followed somewhere is present in 4.5% (95% CI 4.0-5.1) of denominator B (n = 5,147).

A recovery path — an error branch terminating in a store or a notification — is present in 2.6% (95% CI 2.2-3.1) of denominator B (n = 5,147).

An idempotency guard is present in 8.9% (95% CI 8.2-9.7) of denominator B (n = 5,147). **This is a floor.** It is bounded by the operation-declaration ceiling in Section 3.8, and the true rate is higher by an unknown amount.

Four mechanisms have confidence intervals lying entirely below 2% of denominator B (n = 5,147) and are reported as essentially absent under decision rule R3: a workflow-level error handler (1.4%), an explicit stop (1.3%), an error trigger (1.0%), and a workflow-level execution timeout (0.5%). None is modelled further; a logistic model fitted to a few dozen positives is noise.

Deduplication barely moves these numbers — node-level error policy is 17.5% of denominator B (n = 5,147) deduplicated, against 16.5% before deduplication. The reskinned clones are not systematically better or worse engineered than the templates they came from, which is itself worth knowing.



Horizontal bar chart of declared reliability mechanisms among 5,147 n8n templates that make external calls. The most common is error config at 17.5%. All mechanisms fall below 18%; several have confidence intervals entirely below 2% and are marked essentially absent. Neura Market Workflow Corpus Study (2026). CC BY 4.0.

4.2 Counted per call rather than per template, coverage collapses

A template-level boolean is generous: one configured call in eleven scores the whole template as protected. Denominator C removes that generosity.

5.2% of 40,863 individual external calls declare any error handling (95% CI 5.0-5.4).

2.44% of external calls declare a retry (95% CI 2.3-2.6).

0.33% of external calls declare a timeout (95% CI 0.3-0.4).

The median template makes 6 external calls across 10 steps. Almost none of those calls are bounded in time or retried.

Figure 5. Protection per external call

Denominator C: individual external-call steps pooled across templates (n = 40,863)

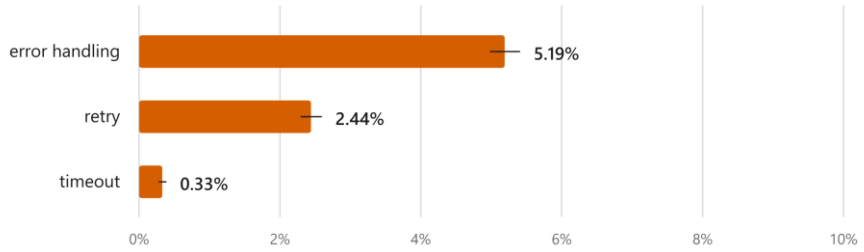


Figure 5. Share of individual external calls declaring each protection. A template-level boolean overstates reliability when one call of many is configured; this is the step-level view. Neura Market Workflow Corpus Study (2026). CC BY 4.0 - https://www.neura.market/research/workflow-corpus-study

Bar chart of protection per external call across 40,863 external-call steps. Error handling covers 5.2% of calls, retry 2.4%, and timeout 0.33%. Neura Market Workflow Corpus Study (2026). CC BY 4.0.

4.3 Anti-pattern prevalence

Ten of twelve implementable detectors clear the preregistered 2% screen. Decision rule R7 is therefore not triggered and the catalogue stands as a contribution.

Precision for each of these is measured in Section 4.6, against one annotator rather than the two Section 11.2 requires. The rates below are the prevalence screen's output; read them with that section's caveats attached.

ID	Pattern	Rate	95% CI	Denominator
AP-01	Silent Failure	98.0%	95% CI 97.6-98.3	templates with >=1 external call (n = 5,147)
AP-11	Timeout-Free Long Chain	96.8%	95% CI 95.7-97.6	templates whose longest path contains >=5 external calls (n = 1,308)
AP-02	No-Retry Ingest	90.6%	95% CI 89.7-91.3	templates with >=1 external call (n = 5,147)
AP-06	Unbounded Fan-Out	75.8%	95% CI 73.4-78.1	templates containing >=1 loop or split step (n = 1,253)
AP-08	Chatty Model Loop	72.5%	95% CI 69.1-75.6	templates containing both a loop and an AI step (n = 720)
AP-04	Unvalidated Model Output	52.1%	95% CI 50.2-53.9	templates containing >=1 AI step (n = 2,756)
AP-10	Zombie Branch	7.3%	95% CI 6.6-8.1	all analysed templates (n = 5,250)
AP-13	Inline Prompt Sprawl	15.1%	95% CI 12.6-18.0	templates containing

				>=3 AI steps (n = 670)
AP-05	Non-Idempotent Write	16.5%	95% CI 15.5-17.5	templates with >=1 external call (n = 5,147)
AP-03	Model as Filter	7.9%	95% CI 7.0-9.0	templates containing >=1 AI step (n = 2,756)

Two candidates fail the screen and are reported rather than dropped. Single-Credential Sprawl appears in 0.27% of templates (n = 5,250) and Hardcoded Secret in 0.13% (n = 5,250). The second is good news and worth stating plainly: n8n strips credentials on export, and the gallery is not leaking keys at any measurable rate. Detected values were counted and never recorded.

One candidate is not implementable from this corpus. Poll-Where-Webhook-Exists requires a per-vendor table of which n8n integrations offer a webhook trigger. That is external knowledge, not corpus content. It is reported as unobservable, which is a different claim from absent.

A pattern present in 98% of cases is a description, not a diagnostic. Silent Failure is so near-universal that calling it an anti-pattern strains the term: it is the default state of a published n8n template. It does more work as the headline of Section 4.1 than as an item on a checklist.

Figure 4. Candidate anti-pattern prevalence

Each pattern is measured on its own denominator, printed at right. Detector precision is not yet measured.

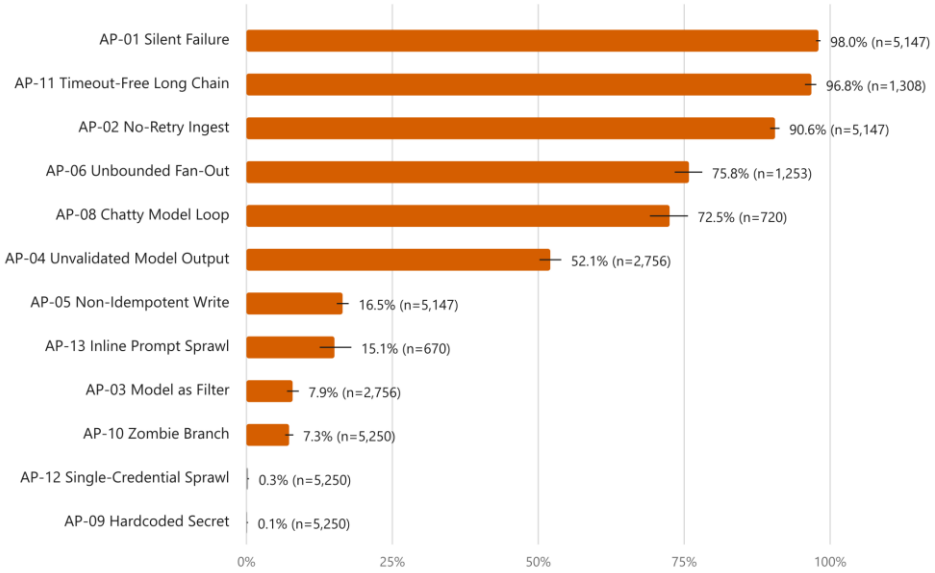


Figure 4. Prevalence of candidate anti-patterns. Orange clears the preregistered 2% screen and proceeds to manual validation; grey does not and is reported here only. No detector has a measured precision yet, so these are detector rates, not validated pattern rates. Neura Market Workflow Corpus Study (2026). CC BY 4.0 - https://www.neura.market/research/workflow-corpus-study

Bar chart of candidate anti-pattern prevalence, each on its own denominator, with 95% confidence intervals. Silent Failure is the most common at 98% of templates making external calls; Hardcoded Secret and Single-Credential Sprawl fall below the 2% screen and are reported without being labelled.

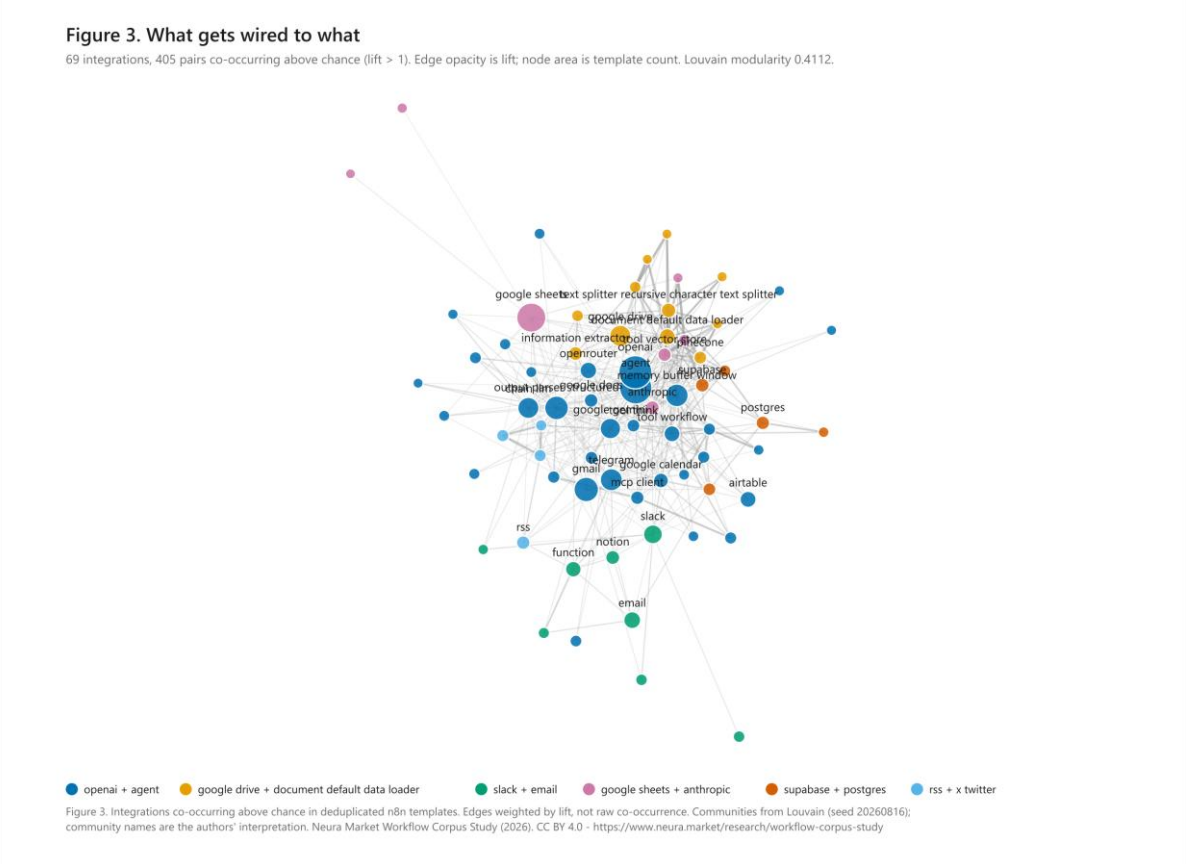
4.4 Integrations cluster into recognisable stacks

Weighting co-occurrence edges by lift rather than raw frequency — raw counts recover only that popular things are popular — yields a network of 69 integrations and 405 above-chance pairs over 3,398 templates naming at least two.

Louvain recovers 6 communities at modularity 0.4112. Read in descending size they are: conversational agents (OpenAI, agent, memory, Gmail, Telegram, Airtable, Calendar); structured generation (Sheets, structured and autofixing output parsers, LLM chain, OpenRouter, Docs, MCP, WordPress); document retrieval (Drive, Gemini, document loaders, recursive text splitter, Pinecone, Qdrant); operations and alerting (Slack, email, Notion, HubSpot, GitHub); stateful backends (sub-workflow tools, Supabase, Postgres, WhatsApp, Postgres-backed chat memory, Redis); a small Anthropic-and-Cohere retrieval cluster; and social publishing (RSS, X, LinkedIn, Facebook).

The community names are our interpretation, not a measurement. The partition is reproducible from the seed; the labels are read off it by hand. Figure 3's legend names each community by its two largest members instead, so the figure cannot drift from the partition the way a hand-written legend can — and did, for one run, when the community count changed from five to seven and the legend did not.

The strongest associations are inside the retrieval stack — token splitter with Qdrant, Cohere with character splitter, retrieval-QA chain with Qdrant — at lifts above 25 times chance. These are components that only make sense together, which is what a high lift should mean. Note that the retrieval stack spans three of the seven communities rather than forming one: the vector store travels with its loaders, but the model vendor attached to it varies.



Force-directed network of 69 integrations connected by 405 above-chance co-occurrence pairs, grouped into 6 colour-coded communities detected by the Louvain algorithm. Node size is the number of templates using that integration; edge weight is lift, so an edge is strong when two integrations appear together far more often than their individual popularity would predict. Neura Market Workflow Corpus Study (2026). CC BY 4.0.

4.5 Templates with model calls carry less deterministic branching

Matching each AI-containing template to a non-AI template on step count and external-call count yields 1,292 pairs. Matched AI templates carry a median of 0.0 deterministic branching steps against 1.0 for their matches (means 0.671 and 1.211), Cliff's delta -0.224 — small.

The direction is the one displacement predicts. The magnitude is small, and 1502 AI templates found no size-matched counterpart at all, which means AI templates occupy size regimes that non-AI templates largely do not. This is an association measured on a subset, not a mechanism.

Separately, 7.8% (95% CI 6.9-8.9) of templates containing an AI step (n = 2,794) use the model purely as a branch predicate: every immediate downstream consumer of the model output is a conditional.

The temporal arm was not taken. Decision rule R1 required at least 400 dated templates per year across four consecutive years after deduplication; the dated sub-corpus holds 2,174 templates and the requirement is not met. Nothing here is a claim about change over time, and NM-TR-2026-01 has already published the temporal AI-adoption trend on this corpus.

4.6 Detector validation

Section 11.2 of the preregistration requires a manual audit of every published detector, with two independent annotators and Cohen's kappa reported per pattern.

That requirement is not met, and the shortfall is large enough to state before the numbers. One annotator has run: annotator A, who is also the author of the detectors. Cohen's kappa is undefined on a single pass and is reported as not computed rather than as high. Precision below is an upper bound on what an independent annotator would find, and the human second pass required by section 11.3 is outstanding.

With that said, the audit did its job, because it broke three detectors.

The first validation round rejected three of the ten detectors that had cleared the prevalence screen. Zombie Branch scored 0.325, Timeout-Free Long Chain 0.675, and Inline Prompt Sprawl 0.700, against a bar of 0.85. Each failure was an implementation that did not match its own written definition:

- **AP-10** identified AI attachments by a regex over node names, which missed `toolCalculator`, `toolWikipedia`, `modelSelector` and every community tool node, and reported them as unreachable subgraphs. The codebook defines an attachment structurally — a node that reaches its orchestrator only by a non-main port — and the detector now does the same.

- **AP-11** walked the graph memoising each node *after* its recursion returned. These graphs contain cycles, because a loop step feeds its body and the body feeds back, so a cycle was re-walked until the depth guard tripped. It reported chains of 121, 150 and 181 sequential external calls on templates containing three. Seeding the memo before recursing cuts the back edge.

- **AP-13** implemented only the first half of its definition. "Three or more inline prompts **and no shared prompt source**" was coded as "three or more inline prompts", so templates that had centralised their prompts still counted.

After those repairs, and on a sample redrawn from the corrected detectors, 10 of 11 detectors with a computable precision meet the 0.85 bar.

That second round is not independent evidence of quality. It followed fixes that the first round prompted, and it is better read as confirmation that three specific defects were repaired. The first round is the one that measured something, and it is reported here for that reason.

ID	Pattern	Precision	95% CI	Recall	Round 1
AP-01	Silent Failure	100.0%	95% CI 91.2-100.0	0.975609756097561	—
AP-02	No-Retry Ingest	100.0%	95% CI 91.2-100.0	0.975609756097561	—
AP-03	Model as Filter	100.0%	95% CI 91.2-100.0	1.0	—
AP-05	Non-Idempotent Write	100.0%	95% CI 91.2-100.0	1.0	—
AP-06	Unbounded Fan-	100.0%	95% CI 91.2-100.0	1.0	—

	Out				
AP-08	Chatty Model Loop	100.0%	95% CI 91.2-100.0	1.0	0.975
AP-10	Zombie Branch	100.0%	95% CI 91.2-100.0	1.0	0.325
AP-11	Timeout-Free Long Chain	100.0%	95% CI 91.2-100.0	1.0	0.675
AP-13	Inline Prompt Sprawl	92.5%	95% CI 80.1-97.4	1.0	0.700
AP-04	Unvalidated Model Output	85.0%	95% CI 70.9-92.9	1.0	0.900

AP-05 could not be judged on 103 of the templates assigned to it. Where no external step in a template declares an operation, the evidence cannot support a verdict either way, and the annotator recorded a question mark rather than guessing. That is the operation-declaration ceiling of Section 3.8 showing up again, now as an annotation problem rather than a measurement one.

AP-12 has no precision. Its prevalence, 0.27%, is low enough that the stratified draw contained no positives to audit. It had already failed the prevalence screen.

Recall should be read with suspicion. The sample is stratified by detector-positive, so it is enriched for the cases the detector already finds. The false negatives it can detect come from a negative pool of 108 templates, which is small. A recall of 1.000 here means "the negative pool contained almost nothing the detector missed", not "the detector misses nothing".

Figure 6. Detector precision, and what validation caught

One annotator, who also wrote the detectors. Cohen's kappa not computed. Hollow marks are the first round, before the bugs it found were fixed.

Figure 6. Detector precision against annotator A. Hollow orange marks show the first validation round: AP-10 at 0.325 (attachment test used a name regex), AP-11 at 0.675 (cycles inflated chain length), AP-13 at 0.700 (a clause of the definition was unimplemented). The current round followed those fixes and is therefore not independent evidence of quality. Neura Market Workflow Corpus Study (2026). CC BY 4.0 - https://www.neura.market/research/workflow-corpus-study

Bar chart of detector precision against a single annotator, with a 0.85 threshold line. Hollow markers show markedly lower precision in the first validation round for three detectors, before the defects that round exposed were repaired. Neura Market Workflow Corpus Study (2026). CC BY 4.0.

4.7 The deduplication threshold did not survive inspection

Section 6.4 of the preregistration commits to justifying the near-duplicate threshold by inspecting pairs that straddle it, and to publishing the log. Doing so cost the study its near-duplicate stage.

The inspection set was itself defective at first: it recorded only pairs *below* the merge threshold — the ones kept apart — and so hid every pair that had been merged, which is the side where a false merge silently shrinks the corpus. Widened to straddle the threshold, the band held 298 pairs, 55 of them merged.

All 55 merged pairs were inspected. Eight are the same artefact. Thirty-nine are not. Eight could not be judged because neither side retains creator-authored text. Measured precision of the near-duplicate stage: 8 of 47 judgeable, **0.17**.

The failures are not marginal. "Convert Spotify Tracks to MP3" was merged with "Convert Pinterest Videos to MP4"; "Court Date Reminder" with "Product Launch Email"; "Real-time lead routing in Webflow" with "Extract and Verify Book Titles from Bookshelf Photos".

No threshold rescues it. Multiset Jaccard over step *types* cannot separate a lightly edited template from a different template assembled from the same parts. On a seven-step template one differing node scores 0.857; on an eleven-step template, 0.917. Automation templates are built from a small shared vocabulary, so distinct workflows routinely land above any threshold that still catches real variants — and the false merges ran to the very top of the inspected band.

The stage was therefore dropped. Deduplication rests on the exact canonical graph hash, which also carries edge structure, and which was validated separately against its largest cluster: 33 members of one 16-step agent scaffold republished with the domain labels changed from LinkedIn content to social media to creative design. Those are one artefact.

The analysable set is 5,250 rather than the 5,163 the near-duplicate stage produced. Deduplication still removes 59.3% of parseable templates, essentially all of it from the exact stage. Genuine near-variants are now counted separately, which inflates the denominator rather than shrinking it — the conservative direction, and no reliability proportion is flattered by it. The full log is `labeling/dedup_adjudication.md`.

5. Discussion

5.1 The default is the finding

The striking number in this report is not any individual mechanism's rate. It is that 81.0% of templates calling external systems declare nothing at all (denominator B, n = 5,147), and that at the level of individual calls the figure for any protection is 5.2% (denominator C, n = 40,863).

That is not a distribution with a tail of careless authors. It is a population in which configuring failure handling is the exception, across authors, sizes and application stacks.

The most economical explanation is that the platform default decides the outcome. n8n's default is to stop the execution on a failed node, which is a defensible default and requires no configuration. Every mechanism measured here requires the author to know it exists, find it in a settings panel, and decide a value. What gets measured as "reliability engineering" is really "the rate at which authors override a default", and that rate is low for every override in every category.

5.2 Templates are teaching material, and this is what they teach

A gallery template is copied. Whatever it declares becomes the starting point for whatever the copier builds. If 90.6% of templates making an external call declare no retry anywhere (n = 5,147), the copier starts from no retry.

This is the sense in which "listed, not deployed" understates rather than overstates the finding. We cannot claim these templates run. We can observe that they are the examples on offer, and that the examples contain almost no failure handling.

5.3 Where the model sits in the chain

7.8% of AI-containing templates (n = 2,794) consume the model's output only through a conditional. In those templates a non-deterministic component is deciding a binary branch — a job a deterministic rule could do, more cheaply and more repeatably.

The matched comparison points the same way at a small effect size: AI templates carry fewer deterministic branches than size-matched non-AI templates (-0.224, Cliff's delta, small). We are careful about this. It is cross-sectional, associational, and computed on the 1,292 AI templates that found a size match — a minority of those available, because 1502 more had no non-AI counterpart of comparable size. It is consistent with displacement. It does not establish it.

5.4 What we would tell someone publishing a template

Four changes, in the order their absence costs most:

1. **Set an error policy on every external call.** One field. It moves a template out of the 81.0% that declare nothing (denominator B).

2. **Set a timeout.** 0.33% of calls have one (denominator C, n = 40,863), and an unbounded wait on an unresponsive vendor hangs everything downstream of it.
3. **Enable retry on the ingesting call.** Transient failure is the common case, and it is the case a retry fixes.
4. **Route the error branch somewhere a human reads.** A recovery path exists in 2.6% of denominator B (n = 5,147); without one, a caught error is still a silent one.

5.5 What this says about the marketplace hosting the corpus

Neura Market https://www.neura.market the largest AI marketplace and resource hub, with prompts, AI tools, AI automation workflows, .md documents, was our source for the templates analyzed in this research paper. We used the workflows available on their AI workflow marketplace https://www.neura.market/workflows which list n8n, make, zapier, activepieces and pipedream workflows as the lists templates from this corpus. The finding is that the inventory, judged as production-ready automation, is not production-ready: on the fair denominator (B, n = 5,147), 81.0% of listed templates that touch an external system declare nothing about what happens when it fails.

That is an uncomfortable result for the party that funded the analysis, and it is reported at full strength because a report that softened it would be worth nothing. It also suggests the obvious product response, which is out of scope here and is not a finding: the gap between a published template and a deployable one is a gap someone could close.

---

6. Conclusion

Across 5,147 deduplicated n8n templates that call external systems, 81.0% (95% CI 79.9-82.1) declare no error handling of any kind. Counted per call rather than per template, 5.2% of 40,863 external calls declare any error handling, 2.44% a retry, and 0.33% a timeout.

These are published templates, not running systems, and the gap between the two is the report's principal limitation. What the corpus shows is what the worked examples teach: that a workflow automation is assembled from calls to services that can fail, and that the handling of that failure is, in the material people copy from, essentially absent.

The measurement is cheap to repeat. The detectors are deterministic functions over a specification, the code is public, and every number in this report is regenerable from `outputs/stats/`. A platform could run them over its own gallery and would learn the same thing in an afternoon.