A RESOURCE FOR DESIGN SYSTEM LEADS AND THEIR TEAMS

Automation decision framework.

Automate, assist or avoid, decided with a rubric you can argue with.

Score it. Veto it. Gate it. Keep people able to do it.

24	05	13
checks, scoring to review	criteria, scored 1 to 5	worked design-system tasks

Automation is a level, not a switch.

Parasuraman, Sheridan and Wickens described automation as a scale of levels, from a computer that offers no assistance to one that acts without telling anyone, applied separately to gathering information, analysing it, deciding and acting [S01]. Their criteria for choosing a level still hold: how reliable the automation is, what a wrong outcome costs, and what it does to the people who remain responsible.

Design-system teams now make that choice weekly, usually by gut. This guide turns it into five scores: frequency, verifiability, reversibility, blast radius and judgment. Three vetoes send a task to people however well it scores; five floors decide what may run on its own; everything else is assisted, with a model drafting or suggesting and a named person deciding.

The kit has the rubric as JSON, a scorer and 13 scored tasks from the same Acme design system as Field Guides 02 to 04, 19 and 20. The thresholds are a starting point calibrated on those examples, not a research result. Change them, and write down why.

Version 1.0 / Sources checked 24 September 2026
Field Guide 21 of the Design × AI series. Buckets map to the levels of automation in Parasuraman, Sheridan and Wickens (2000). Pairs with Field Guide 19 (doc drafting, an assist task) and 20 (the review gate). Sources checked 24 September 2026.

Practical guidance, not a standard. The research gives the dimensions; the numbers in the rubric are the author's judgment. Prepared with AI assistance and edited by hand.

START HERE

Pick your route, then read the labels.

Score one task you automated recently and one you are about to. The first tells you whether you trust the rubric; the second is why you have it.

Before automating something

Score it with two people who do the task today, run `score-task.mjs --explain` and fill in `templates/decision-record.md`.

Auditing what already runs

Score every existing automation. Anything that lands in avoid, or in assist without a named reviewer, gets a decision record this month.

Arguing about a task

Disagree on the scores, not the bucket. The rationale field per criterion is where the argument belongs.

Setting up the gate

Section 04 lists what an automate or assist decision needs before it ships. Field Guides 19 and 20 are two worked gates.

Read the labels before the checks.

Label	Meaning
`RUBRIC`	Encoded in `rubric.json` and applied by `score-task.mjs`.
`SCRIPT`	Checked by a kit script or CI.
`REVIEW`	Needs a person's judgment and a record of it.
`PRACTICE`	A working method with a review signal rather than a hard gate.

Bucket	Who acts	Levels of automation [S01]
Automate	The system acts; a gate checks every output; a person can veto or is informed	6 to 7
Assist (draft)	The model produces it; a person approves before it lands	5
Assist (suggest)	The model offers options; a person chooses	3 to 4
Avoid	People do it; a model may gather information for them	1 to 2

Levels 8 to 10, where the system informs people only if asked or not at all, have no bucket here.

SECTION 01

Score five things, with reasons.

Each criterion is scored 1 to 5 against written anchors (Appendix A). A score without a sentence of reason cannot be reviewed.

Suggested owners: Design-system lead + two people who do the task

- [ ] **Frequency: how often it happens**

  R01 RUBRIC

  1 is once a year or less, 5 is daily or every pull request. Rare tasks rarely repay the cost of building and watching automation.

  Evidence: `frequency` score with a reason; automate needs 3 or more.

  Recommended practice; the essay's first axis is repetitiveness. S09.

- [ ] **Verifiability: how cheaply the output can be checked**

  R02 RUBRIC

  1 is only taste or time can judge it; 5 is a machine check proves it. This is the reliability of the automation as you can observe it.

  Evidence: `verifiability` score naming the check; automate needs 4 or more.

  PSW: automation reliability as an evaluative criterion. S01. Anthropic: code is verifiable through tests. S04.

- [ ] **Reversibility: how easily a wrong output is undone**

  R03 RUBRIC

  1 is published, sent, deleted or signed; 5 is a draft nobody has seen. npm never lets a version number be reused, so a publish scores 1.

  Evidence: `reversibility` score; automate needs 4 or more.

  PSW: cost of decision and action outcomes. S01. npm unpublish policy. S08.

- [ ] **Blast radius: who a wrong output reaches**

  R04 RUBRIC

  1 is one file or person; 4 is every consumer of the design system; 5 is end users, customers or legal commitments.

  Evidence: `blastRadius` score; automate needs 3 or less.

  PSW: cost of outcomes. S01.

- [ ] **Judgment: how much a good output depends on it**

  R05 RUBRIC

  1 is a rule with one right answer; 3 is trade-offs inside known patterns; 5 is when the judgment is the job: values, strategy, accountability.

  Evidence: `judgment` score; automate needs 2 or less.

  Bainbridge: people are left with what designers cannot automate. S02.

Argue about the scores. The bucket follows.

SECTION 02

Apply the rules in order.

Vetoes first, then floors, then the score. An average never overrides a veto: the npm publish example scores 2.80, above an assist task, and is still avoid.

Suggested owners: Design-system lead

- [ ] **Three vetoes send a task to people**

  R06 RUBRIC

  Judgment 5; reversibility 1 with blast radius 4 or more; verifiability 1 with blast radius 3 or more. Any one of them means avoid.

  Evidence: `vetoes` in `rubric.json`; `--explain` names the veto.

  PSW: high levels of decision and action automation only for low-risk situations. S01.

- [ ] **Automate only when all five floors hold**

  R07 RUBRIC

  Frequency 3 or more, verifiability 4 or more, reversibility 4 or more, blast radius 3 or less, judgment 2 or less. One miss and the task is assisted.

  Evidence: `automateFloors`; `--explain` lists every failed floor.

  Anthropic: agents fit tasks with clear success criteria and feedback loops. S04.

- [ ] **Everything else is assist, unless the score is below 2.5**

  R08 RUBRIC

  The score is the mean of the five criteria with blast radius and judgment inverted. Below 2.5, a task is rare, hard to check and judgment-heavy enough that a model adds more review than it saves.

  Evidence: `avoidBelowScore`.

  The threshold is calibrated on the worked examples, not measured.

- [ ] **Assist has two modes, set by judgment**

  R09 RUBRIC

  Judgment 4 or more: the model suggests options and a person chooses. Below that: the model drafts and a person approves. Doc drafting (Field Guide 19) is draft; naming is suggest.

  Evidence: `mode` in the scorer output.

  PSW levels 3 to 4 and level 5. S01.

- [ ] **Prefer a script when the answer is unique**

  R10 RUBRIC PRACTICE

  Verifiability 5 and judgment 1 means a deterministic script can do the job. Regenerating the API table needs no model at all.

  Evidence: The scorer marks it `automate (script)`.

  Anthropic: find the simplest solution and add complexity only when needed. S04.

Frequency makes automation tempting. Reversibility and blast radius decide whether it is allowed.

SECTION 03

Score the stages, then keep scoring.

One task can hold four decisions. The accessibility audit example scores automate for scanning and avoid for declaring conformance.

Suggested owners: Design-system lead + task owner

- [ ] **Score the four stages separately**

  R11 RUBRIC

  Information acquisition, analysis, decision and action can each sit at a different level. Add a `stages` array to the task and the scorer rates each one.

  Evidence: The a11y-audit example: 4.80 automate, 3.80 draft, 2.80 suggest, 2.00 avoid.

  PSW: the four classes of functions, automated to different degrees. S01.

- [ ] **Automated gathering is not automated deciding**

  R12 REVIEW

  An agent that collects every axe result has automated acquisition. Whoever reads its summary is still deciding, and needs the evidence behind it.

  Evidence: The decision record names who decides at each stage.

  S01. NIST: human-AI configurations span fully autonomous to fully manual. S06.

- [ ] **Close calls are discussed and recorded**

  R13 REVIEW

  Within 0.2 of a threshold, or one point from a floor, write down what you discussed and which side you chose. The naming example sits at 2.60, 0.1 above avoid.

  Evidence: The close-call field in the decision record.

  Recommended practice.

- [ ] **Rescore when anything moves**

  R14 PRACTICE

  A new model, a new tool version, inputs that change shape, an incident caused by the automation, or six months passing. Verifiability in particular changes with the model.

  Evidence: Next review date in the record; `--check` in CI against `expected`.

  Recommended practice.

- [ ] **Keep the expected bucket in version control**

  R15 SCRIPT

  Each task records the bucket you agreed. `score-task.mjs --check` fails when a rubric edit silently moves a task, so threshold changes are reviewed like code.

  Evidence: `node scripts/score-task.mjs examples/tasks.json --check` exits 0.

  Kit script.

The bucket is a decision with a date on it.

SECTION 04

Build the gate before the automation.

The essay behind this guide puts it plainly: every AI output needs a gate [S09]. An automate or assist decision is incomplete until these five exist.

Suggested owners: Task owner + engineering lead

- [ ] **The check exists before the automation does**

  R16 SCRIPT

  If you cannot name the machine check or the review step that catches a wrong output, verifiability is not 4, whatever the demo showed.

  Evidence: The gate field in the decision record names a command or a reviewer.

  Claude Code docs: if you cannot verify it, do not ship it. S07.

- [ ] **Every output passes the gate; people sample**

  R17 SCRIPT REVIEW

  Machine checks run on every output. A person reviews a stated sample, and all outputs the checks warn about.

  Evidence: Gate logs; the sample size in the record.

  Anthropic: sandboxed testing and guardrails for autonomous systems. S04.

- [ ] **Rollback is tested, not planned**

  R18 PRACTICE

  Detecting a bad output, reverting it and telling the people it reached. Run it once before relying on it.

  Evidence: Last rollback test date in the record.

  Recommended practice; the essay's rollback plan. S09.

- [ ] **There is an owner and an off switch**

  R19 PRACTICE

  A named person who can stop the automation, and a way to stop it that does not need the person who built it.

  Evidence: Owner and off switch in the record.

  NIST AI RMF: oversight roles are defined and documented. S06.

- [ ] **Time saved is net of review**

  R20 REVIEW

  Count the minutes spent checking, fixing and reverting, not only the minutes the model saved. The essay's 200 passing tests that tested nothing cost two sprints.

  Evidence: Before and after timings that include review.

  S09.

An automation without a gate is a guess that runs on a schedule.

SECTION 05

Keep the people able to take over.

The more reliable the automation, the less practice people get at the task it replaced. Forty years of research on automation says to plan for that.

Suggested owners: Design-system lead

- [ ] **Reviewers show evidence, not approval**

  R21 REVIEW

  An approve button invites automation bias: using the aid instead of checking. The review step asks for the evidence looked at, such as the failing test or the changed story.

  Evidence: Review comments cite evidence.

  Parasuraman and Manzey: automation bias and complacency, not fixed by training alone. S03.

- [ ] **People still do the task by hand, sometimes**

  R22 PRACTICE

  Rotate who writes a component page or reviews a pull request without the tool, so the skill exists when the tool is wrong.

  Evidence: A named rotation in the record.

  Bainbridge: skills deteriorate when not used. S02.

- [ ] **Trust growing is a signal to look again**

  R23 REVIEW

  Experienced users approve automation more often. That is expected, and it is also when a drop in verifiability goes unnoticed.

  Evidence: Rescore when approval rates rise and review comments thin out.

  Anthropic: experienced Claude Code users auto-approve more often. S05.

- [ ] **The team can see every decision**

  R24 PRACTICE

  Publish the decision records where the team works. People who know a task is assisted review its output differently from people who assume a person wrote it.

  Evidence: One index of decision records per team.

  NIST AI RMF: roles for human-AI configurations and oversight are defined. S06.

Automate the task. Never automate away the people who can do it.

APPENDIX A

The anchors, from rubric.json.

Score against these words, not against how the task feels. Blast radius and judgment count against automation; the other three count for it.

Criterion	1	3	5
Frequency	Once, or less than once a year	Monthly	Daily, or every pull request
Verifiability	Only taste or long-term outcomes judge it	A person checks it against a spec in minutes	A machine check proves it
Reversibility	Cannot be undone: published, sent, signed	A revert others notice	A draft nobody else has seen
Blast radius	One person or file	One product	End users, customers, legal commitments
Judgment	A rule with one right answer	Trade-offs within known patterns	The judgment is the job

Levels 2 and 4 are in the file. The essay's three axes map onto these: repetitiveness is frequency, definition quality is verifiability, stakes are reversibility and blast radius.

rubric.json (excerpt)

```
"vetoes": [
  { "id": "judgment-is-the-job",
    "when": { "judgment": { "gte": 5 } } },
  { "id": "irreversible-and-wide",
    "when": { "reversibility": { "lte": 1 }, "blastRadius": { "gte": 4 } } },
  { "id": "unverifiable-and-wide",
    "when": { "verifiability": { "lte": 1 }, "blastRadius": { "gte": 3 } } }
],
"automateFloors": {
  "frequency": { "gte": 3 }, "verifiability": { "gte": 4 },
  "reversibility": { "gte": 4 }, "blastRadius": { "lte": 3 },
  "judgment": { "lte": 2 }
},
"avoidBelowScore": 2.5
```

Each veto also carries a reason, printed by `--explain`.

APPENDIX B

Thirteen tasks, scored.

Output from the kit. Scores are the author's, each with a reason in `examples/tasks.json`; the buckets are computed.

terminal

```
$ node scripts/score-task.mjs examples/tasks.json --check
Task                                              F V R B J  score  bucket
Regenerate the API table from the contract        5 5 4 2 1   4.60  automate (script)
Flag hallucinated imports and invented props      5 4 5 1 2   4.60  automate
Rewrite raw hex values to their 1:1 semantic token  4 4 4 3 2   3.80  automate
Build the changelog from conventional commits     4 4 4 3 1   4.00  automate
Draft a component documentation page              3 3 5 3 3   3.40  assist (draft)
Propose fixes for axe violations in a component   4 3 4 3 3   3.40  assist (draft)
Generate unit tests for an existing component     4 2 5 2 3   3.60  assist (draft)
Name a new component and its props                2 2 4 3 4   2.60  assist (suggest)
Decide on a breaking change and a major release   1 2 1 4 4   1.60  avoid
Deprecate a component used across products        2 2 2 4 5   1.80  avoid
Sign the accessibility conformance report         1 2 2 5 5   1.40  avoid
Let an agent publish the package to npm           4 3 1 4 2   2.80  avoid
Audit a new component for accessibility           3 2 4 3 4   2.80  assist (suggest)
  acquisition: Run axe on every story                         4.80  automate
  analysis: Group and explain findings                        3.80  assist (draft)
  decision: Choose the fix for each finding                   2.80  assist (suggest)
  action: Declare the component conformant                    2.00  avoid
13 task(s): 4 automate, 5 assist, 4 avoid
```

`--check` exits 0: every bucket matches the expected one recorded in the file.

The results agree with the essay on its examples: changelogs and prop extraction are automated, documentation and test generation are generated and validated, naming and API options are assisted with a person deciding, breaking changes stay with people [S09]. Two refinements come from the vetoes. A changelog built from commit metadata by a script is automate; release notes written by a model would score lower on verifiability. And a frequent, low-judgment task such as publishing is still avoid for an agent, because nothing takes a published version back.

KEEP WITH THE AUTOMATION

Leave a decision record.

The kit's `templates/decision-record.md` has the full form. These are the fields that matter most when something goes wrong.

Task / owner	**Scores with one reason each**
Bucket, mode and level of automation	**Close-call discussion (if within 0.2)**
Machine gate (command or check)	**Human review and sample size**
Off switch and who can use it	**Rollback, last tested**
Who still does it by hand	**Next review date**

> **Record the no as well.**
> A task scored avoid is a decision too. Writing it down stops the same automation being proposed every quarter.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. The research papers supply the dimensions and the human factors; the rubric's numbers are the author's and are labelled as such.

S01 / Parasuraman, Sheridan and Wickens, A model for types and levels of human interaction with automation (2000)
Four function classes, the ten-level scale (after Sheridan and Verplank 1978), and reliability and outcome cost as criteria. IEEE SMC Part A 30(3).
https://doi.org/10.1109/3468.844354

S02 / Bainbridge, Ironies of Automation (1983)
Operators keep the tasks designers cannot automate; skills deteriorate without use. Automatica 19(6).
https://doi.org/10.1016/0005-1098(83)90046-8

S03 / Parasuraman and Manzey, Complacency and bias in human use of automation (2010)
Automation bias and complacency; neither is overcome by training or practice alone. Human Factors 52(3).
https://doi.org/10.1177/0018720810376055

S04 / Anthropic, Building effective agents (December 2024)
Simplest solution first; workflows and agents; clear success criteria, feedback loops, tests, sandboxes and guardrails.
https://www.anthropic.com/engineering/building-effective-agents

S05 / Anthropic, Measuring AI agent autonomy in practice (February 2026)
Experienced Claude Code users auto-approve more often and interrupt more often.
https://www.anthropic.com/research/measuring-agent-autonomy

S06 / NIST AI Risk Management Framework 1.0 (AI 100-1)
Human-AI configurations from fully autonomous to fully manual; defined oversight roles.
https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf

S07 / Claude Code, best practices
Give the agent a way to verify its work; if you cannot verify it, do not ship it.
https://code.claude.com/docs/en/best-practices

S08 / npm, unpublish policy
Limited unpublishing, and a used package@version can never be used again.
https://docs.npmjs.com/policies/unpublish

S09 / Petri Lahdelma, AI in design systems: what actually works
The essay's decision flow, its examples, the rollback plan and the 200-test failure story.
https://petrilahdelma.com/writing/ai-design-systems-practical

Maintenance: rescore the worked examples when a new model generation changes what is verifiable, and run `score-task.mjs --check` after any rubric edit. Update the PDF, HTML, Markdown and JSON together.