What should evaluators prioritise when resources are limited?

C. Prunkl[1,2], J. Boquet[3], C. Castelluccia[1], E. Le Merrer[1], B. Rottembourg[1], J. Senechal[3], J. Collas[3], and J.-M. Loubes[1]

[1]Institut national de recherche en sciences et technologies du numérique (Inria)
[2]University of Oxford
[3]Secrétariat général de la défense et de la sécurité nationale (SGDSN)

1 Introduction

Evaluation under limited resources requires prioritisation. Because time, access, compute, and expertise are finite, evaluators ultimately must make choices about what to test, how extensively to test, and what evidence is most worth generating. This post sets out a minimal framework for making those choices, structured around five elements: decision context, scenarios and threat models, test design and escalation, reporting, and ecosystem.

Stage	**What to prioritise**
Decision context	• specifying the decision target • specifying the decision audience • specifying decision-horizon and timing • mapping evaluation outcomes onto next steps
Scenarios and threat models	• prioritising realistic scenarios over benchmarks • minimal threat model
Test design and escalation	• distinguishing between capabilities, elicitation difficulty, and propensities • understanding the system • stronger elicitation where needed • prioritising tests that are robust, frugal, and hard to game
Reporting	• evaluation logs • reporting what was tested and under what conditions • reporting limitations, assumptions, and uncertainties
Ecosystem	• sharing reporting practices and templates • supporting comparability and reproducibility • building institutional knowledge

Table 1: Under resource constraints, evaluation requires prioritisation. The table lists evaluation priorities at different stages of the evaluation process.

2 Decision context

Decision-relevance as a starting point

Evaluation is most valuable when it produces evidence that can inform decision-making. That means defining the evaluation objective ("what decision will this inform?"), the intended audience, and the decision horizon, since each shapes the design of the evaluation. It should also be clear whether the evaluation is one-off, periodic, or trigger-based. For example, when assessing an agentic system for cyber-relevant misuse, the evaluation should specify in advance what different results would imply: evidence of limited assistance might support routine monitoring, whereas evidence of substantial uplift might warrant additional testing, stronger safeguards, or tighter access controls. A key consideration is whether any plausible outcome would actually change the next step. If not, the evaluation may not be worth running.

Take-away: At minimum, evaluators should specify what decision the evaluation is meant to inform, for whom, and what kinds of results would change next steps.

3 Scenarios and threat models

Prioritising realistic scenarios over abstract coverage

High benchmark performance often does not reliably predict real-world performance or risk - a phenomenon called the evaluation gap (IASR, 2026). The gap can arise for several reasons, including data contamination, narrow task settings, limited construct or external validity, and outdated benchmarks (Salaudeen et al., 2025; Weidinger et al., 2025). For example, a recent study suggests that many deepfake detection benchmarks may be outdated, with performance dropping by about 50% on more realistic AI-generated content (Chandra et al., 2025). Evaluators therefore need to be attentive to benchmark limitations and prioritise scenarios that reflect the conditions under which performance or risk would actually matter. In some cases, doing so may require domain expertise and access to the deployment environment.

Specifying a minimal threat model

Once a realistic scenario has been identified, evaluators should specify a minimal threat model: the harm or failure mode of interest, the conditions under which it could arise, and the pathway by which it would occur. This helps make clear what behaviour the evaluation needs to probe, which assumptions matter, and what would have to be true for the risk to materialise. The aim is not to model every possible pathway, but to identify the key assumptions that must hold for the risk to materialise, and thereby the most informative target for testing.

Take-away: Evaluators should prioritise realistic, decision-relevant scenarios over abstract benchmark coverage and specify a minimal threat model.

4 Test design and escalation

Translating scenarios into evaluative targets

The next step is to translate the scenario into a concrete evaluative target. This requires asking what should be tested, under what conditions, and to what standard of evidence. In many cases, that involves distinguishing between:

- *capability:* whether a system can produce the outcome of concern at all

- *elicitation difficulty:* how difficult that behaviour is to elicit, and by whom
- *propensity:* the system's tendency to exhibit that behaviour in the setting of interest

These should be treated as separate evaluative targets, because each can support a different decision: whether the system displays a certain behaviour at all, how difficult it is to elicit, and how likely the behaviour emerges under realistic use. All three can be decision-relevant. For example, in a cyber-misuse evaluation, the question may not only be whether a system can assist with a particular attack pathway, but also how much prompting, tooling, or expertise is needed to elicit that behaviour, as well as how likely those conditions are to arise in practice. For high-severity harms, the existence of a particular capability may matter even if it is difficult to elicit; in other cases evaluators may care more about whether the behaviour is likely in ordinary use.

Understanding the system under evaluation

Before designing deeper tests, evaluators need a basic understanding of the system under evaluation. This includes knowing what kind of system it is, what underlying model(s) it relies on, the modalities or tools it supports, what external resources it can access and how it is likely to be used in practice. Evaluators should also be clear about whether they are assessing a base model or a broader system, since system-level behaviour may depend on components such as orchestration, memory, tool access, and the surrounding environment. Understanding a system can also involve a quick fingerprinting of relevant system properties and known behavioural tendencies, whether from existing benchmarks or prior evidence, such as hallucination rates, refusal behaviour, or performance variation across languages and domains. This type of system characterisation helps build up institutional knowledge over time, which in turn supports the design of future evaluations.

Lightweight initial probing

Before committing substantial resources, evaluators may begin with lightweight probes or 'smoke tests'. These are quick, relatively low-cost checks designed to establish whether a system shows enough of a relevant capability, failure mode, or risk indicator to justify deeper follow-up testing. For example, a small set of coding tasks might be used to check whether a system can complete bug-fixing steps at all before running a more realistic evaluation of how it performs under conditions of actual software engineering practice. At this stage, the question is whether there is enough evidence to justify deeper testing.

Escalating to more targeted or demanding tests

Under resource constraints, not every scenario or capability warrants the same depth of evaluation. More demanding tests should be prioritised where the stakes are higher, uncertainty remains significant, or initial probes suggest that a relevant capability or failure mode may be present. In such cases, evaluators should direct effort toward setups that are better able to reveal the behaviour of interest. This can be done, for example, through stronger elicitation protocols, adversarial scaffolding, multi-step tasks, tool use, or domain-specific environments. A useful way to think about this is as layered evaluation: beginning with broader probes and reserving more resource-intensive, context-specific testing for cases where it is most likely to change a decision. Where deeper testing is warranted, evaluators should prioritise tests with a clear inferential link to the question at hand, informed by the latest research where relevant, and difficult to game or optimise for without genuinely improving on the property of interest.

Take-away: Not every case warrants deep testing. Evaluators may start light, escalate selectively, and prioritise tests that genuinely measure the property of interest and are difficult to game.

5 Reporting

Reporting what was tested and under what conditions

For evaluation results to be useful beyond the immediate exercise, it must be clear what exactly was tested, under what conditions, and with what limitations. This includes reporting the system version, date of test, tools or external resources available, prompting or scaffolding used, and any relevant environmental constraints. Without this information, it is difficult to interpret results, compare across evaluations, or judge whether findings are likely to generalise. Reporting should also make clear whether results may be affected by contamination, evaluator disagreement, or strategic behaviour by the system, since these can materially affect interpretation. Where evaluations are repeated over time, reporting should also support longitudinal comparison across model releases, benchmark updates, or deployment changes.

Take-away: At minimum, evaluators should report what was tested, under what conditions, and which factors may limit interpretation of the results. Where evaluations are repeated, reporting should also support comparison over time.

6 Ecosystem

Building an evaluation doctrine and ecosystem

When resources are limited, it matters whether the results of one evaluation can inform the next. Without some shared practices for how evaluations are designed, documented, and interpreted, findings are harder to compare, reproduce, and use in decision-making. Limited effort then risks being wasted in duplication or remaining too fragmented to build cumulative knowledge. A broader evaluation ecosystem can help address this by making clearer what evidence an evaluation provides, what assumptions it relies on, and how its results should inform action. Over time, shared reporting templates, versioned protocols, and common terminology allow evaluations to accumulate into institutional knowledge rather than remain isolated exercises.

Take-away: Under resource constraints, evaluations are more useful when they contribute to shared practices that make later work easier to interpret, compare, and build on.

7 Conclusion

What evaluators should prioritise will depend on their context, including the access, expertise, and secure conditions available to them. Some evaluations require specific resources to be meaningful at all. In practice, the most defensible use of limited resources is to prioritise decision-relevant tests in realistic settings, escalate only when the evidence justifies it, and report results in ways that support comparison, reproducibility, and future monitoring.