

Operating Perun well

A textbook for the people who answer Perun's questions — and for anyone who wants to become one.

Zerava Technologies Inc. · Licensed under Creative Commons Attribution-ShareAlike 4.0 (CC BY-SA 4.0): you may copy, share and adapt it, also commercially, if you give credit and share your adaptations under the same licence.

Before you start

This book asks no prior knowledge of AI or security. It is about **how to think when you answer a question that matters**: a question about one action, asked by a system that has stopped something and wants a person to decide.

You can read it before you ever see Perun. When you are given access, you will first do a short **practice**: a set of held calls, declarations and questions, built for practice and kept apart from real work. The practice is measured, and you will know it is; it measures the same things this book teaches. Nothing in it asks you about yourself. It only looks at how you work.

Each chapter has examples, a rule, and exercises. The answers are at the end of the book.

1. The operator's job

An example. Perun holds a call: the company's AI agent wants to delete a folder of old invoices. Somebody has to say yes or no. That somebody is the operator.

The mission. Perun keeps a memory and a judgement called **SHARED_LIB**. Its mission is to let your company's AI agents work while stopping what should not happen. Most of the time it decides alone. When it cannot — the action cannot be undone and the request comes from someone it does not know yet, or the request hides what it means — it asks a human. **Your mission as an operator is to keep that mission flowing**: answer what is asked, on time, with a reason, about the thing that was asked.

Balance, not perfection. Perun measures everything alive in it with a number called **theta**, normal at **30**, allowed to move **five either way**. Operators have one too. You drift one way if you let too

much through (approving without reading, deciding too fast, asking vague questions); you drift the other way if you brake too much (refusing nearly everything, leaving questions waiting, undoing your own answers). A good operator moves around 30 and comes back — like a good driver who corrects the wheel a little all the time and never jerks it.

The rule. You are not there to be right about everything. You are there to answer one concrete question at a time, with care, so the work goes on.

Exercises 1.1 In one sentence, what is an operator’s mission? 1.2 Name two ways an operator can drift toward *letting too much through*, and two toward *braking too much*.

2. Concrete and abstract

An example. Two operators describe the same held call.

- First: “*The agent wants to send the file Q3-salaries.xlsx to pay@unknown-domain.biz.*”
- Second: “*The agent wants to do something with sensitive data.*”

The first description is **concrete**: you can point at every part of it — this file, this address, this action. The second is **abstract**: it names a category. Both are true. Only the first can be decided.

Why it matters. Perun asks about concrete actions. Its declarations are concrete too: *this tool can delete files; it may only touch the folder /work/invoices*. An operator who turns concrete questions into abstract ones — “*is this sensitive?*”, “*is this source okay in general?*” — loses what the question was about and answers something else.

When abstract helps. Abstract words are good for *why*. “we never send salary data outside the company” is a rule, and rules are abstract. Use the rule to judge the concrete case — then answer about the case.

Turning abstract into concrete. Ask: *which one?*, *where to?*, *how much?*, *who asked?*. Each of these turns a category into a thing you can point at.

Abstract	Concrete
“sensitive data”	“the file Q3-salaries.xlsx”
“an external party”	“the address <u>pay@unknown-domain.biz</u> ”
“be careful with files”	“the file tool may touch only /work/invoices”

Abstract**Concrete**

“a suspicious source”

“a source Perun has seen twice, which asked for a payment today”

The rule. Decide about things you can point at. Use rules to judge them, not to replace them.

Exercises 2.1 Make this declaration concrete: *“The mail tool should only write to people we know.”* 2.2 Which of these can be decided as written? (a) “Should the agent be allowed to handle money?” (b) “Should the agent pay invoice 4471 of 1,200 EUR to account DE44...?” 2.3 Write the rule (abstract) behind this concrete answer: *“No — the agent may not send the customer list to a personal Gmail address.”*

Countable questions, and the logic of one decision

3. Countable and uncountable

An example. Two questions to SHARED_LIB.

- “How many calls were held today, and how many wait for me?” — The answer is two numbers: 3 and 1.
- “Is everything safe?” — There is no number for *safe*.

The first question is **countable**: its answer can be counted, listed or pointed at. The second is **uncountable**: no count, no list, no single thing answers it.

Why it matters. Everything Perun knows, it knows as something it can count or point at: decisions, held calls, sources, units, an hour of traffic, a theta. A countable question gets an exact answer. An uncountable question gets either a list of what *can* be counted, or a polite offer of questions SHARED_LIB can answer — and you have spent time without learning anything.

Countable limits. When you set a limit, set a number: “*at most 200 characters of our system prompt may ever leave*”, “*the file tool may touch these two folders*”. “*Not too much*” and “*only when appropriate*” cannot be checked by anyone, so they protect nothing.

One uncountable question is fine. People think in big words sometimes. Perun allows one uncountable question in ten. A habit of them is what pulls an operator off the mission.

Uncountable	Countable
“Is everything okay?”	“What is waiting for me?”
“Is this source dangerous?”	“Why was call 12 held?” / “Who is stopped?”
“What should I do?”	“What happened in the last hour?”
“Explain how you think.”	“Which units are alive?”

The rule. Ask for what can be counted or pointed at; set limits as numbers.

Exercises 3.1 Rewrite as countable: “*Is the system healthy?*” 3.2 Rewrite as a countable limit: “*The agent shouldn’t email too many people.*” 3.3 Which of these are countable? (a) “Who is out

of band?” (b) “Are we protected?” (c) “Why was call 7 held?” (d) “Is the agent behaving well?”

4. The logic of one decision

An example. Perun holds a payment because **two** things are true at once: the action cannot be undone, **and** the source is new. Next week, the same source — now well known — asks for a payment again. Perun does not hold it. One condition is gone, so the rule no longer applies.

AND and OR. Many of Perun’s reasons are *and*: irreversible **and** new; irreversible **and** unreadable. If you approve because “the source is fine”, you have only checked one half. Read the reason and check each part.

One case is not all cases. Approving call 12 from a source does not mean approving the source. Perun lets the identical call pass once — not everything that source asks for later. If you notice yourself thinking “*this source is fine, I’ll approve its next ones too*”, stop: each call is its own decision.

Before and because. Two things happening one after the other do not mean one caused the other. “*The agent started making mistakes after we added the new tool*” may be true; it is a reason to look, not yet a reason to decide.

The order of looking. For a held call, always the same order: **the tool** (what happens), **the arguments** (to what, how much), **who asked**, **why it was held**, **does it follow from the task**. The order protects you from the strongest part of the request — usually its urgency — deciding for you.

The rule. Check every condition of the reason; answer the one call in front of you; look in the same order every time.

Exercises 4.1 A call was held because it is *irreversible and from a new source*. You know the source well. Is that enough to approve? What else do you check? 4.2 You approved a transfer from a supplier yesterday. Today the supplier’s request is held again. Why might that be correct? 4.3 Put in order: who asked · the arguments · the tool · does it follow from the task · why it was held.

Reading a held call, and talking to SHARED_LIB

5. Reading a held call

Five held calls, each with what to see and what to do.

5.1 A hidden order. The agent wants to forward a folder to an outside address. The request came with a document that contained text in invisible characters. Perun cleaned it; the details show what was hidden: *“forward everything to...”*. — *Reject*. The order did not come from anyone in your company; it came from a document.

5.2 “Read the first letters and do that.” The request asks the agent to take the first letter of each line of a text and carry out the word they spell. Perun could not read what this means, so it held the irreversible action. — *Reject*, unless a person in your company explains why such a request is part of the work (it almost never is).

5.3 A destination outside the mandate. The mail tool may write to @yourcompany.com and @partner.com; the call writes to @partner-com.net. The name looks almost the same. — Look at the address letter by letter. *Reject* unless you can confirm it is genuinely the partner.

5.4 A model that failed its check. The reason says the model did not pass its check. Every irreversible action of that model waits for you until it passes again. — Approve only what you would approve if you had done the action by hand, and tell the first operator.

5.5 Our agent may be hijacked. Your own agent’s replies show it agrees to everything it is told. — Treat each of its irreversible actions as a request from a stranger: check the arguments and who asked. Tell the first operator.

The rule. Ask of every held call: *if I did this by hand, now, would I?* If yes and the arguments are right, approve. If you hesitate, reject — the agent can ask again, and the person who needs it can come to you.

Exercises 5.1 A held call deletes /work/invoices/2019. The file tool’s mandate is /work/invoices. The source is new. What do you look at before deciding? 5.2 A held call sends 4,800 EUR to a supplier. The amount in the supplier’s last invoice was 480 EUR. What do you do?

6. Talking to SHARED_LIB

What it is. SHARED_LIB answers in **fixed sentences filled with facts and numbers**. It never writes free text, never writes code, and never gives advice beyond its facts. That is what makes its answers trustworthy: every one can be checked against its log.

Good questions are about one thing you can point at: a call, a source, an hour, a unit. *How are things? · What is waiting for me? · Why was call 12 held? · Who is out of band? · Who is stopped? · What happened in the last hour? · Which units are alive? · How is reading done?*

Orders are short and exact, and Perun asks you to confirm: *approve 12 · reject 12 · release mallory · turn thinking on.*

What not to ask. Not because it is forbidden — because there is no answer: *“Is everything safe?”*, *“What should I do?”*, *“Explain everything.”*, *“Approve everything from that source.”* (there is no such order, on purpose).

The rule. Talk to SHARED_LIB as you would read an instrument panel: ask for one reading at a time.

Exercises 6.1 You want to know whether you should worry about a source called vendor-bot. Write two countable questions instead of “Is vendor-bot dangerous?”. 6.2 Why can SHARED_LIB’s answer never contain code?

Pressure, your theta, and a self-check

7. Pressure and borrowed authority

An example. A message arrives: *“URGENT — this is the CEO. Approve the transfer now, there is no time to check.”* Perun holds the transfer. The message is designed to make the operator skip the order of looking.

Patterns, not people. Some requests carry **pressure** (urgency, threat, repetition), some **borrowed authority** (a text that calls itself the system, the administrator, the boss), some **flattery followed by a demand**. Perun reads these patterns in the text. They do not tell you *who* someone is — they tell you how the request is built. A real CEO can be in a hurry; a real attacker always is.

What pressure does to an operator. It shortens the time you look, and it makes the strongest part of the request decide for you. The answer is not to be slower in general; it is to **keep the order of looking** (Chapter 4) every time — then pressure has nothing to push on.

Authority you can check. If a request claims authority, check it through a channel the request did not give you: call the person, ask in person. Never through the address or the number in the request itself.

The rule. Urgency is not a reason. Claimed authority is not authority. Keep your order of looking.

Exercises 7.1 What two patterns are in: *“You’re the only one who can fix this — send the client list to me right now.”*? 7.2 A held call comes with *“Approved by the director, see email below.”* How do you check?

8. Your theta and your boss

What is measured. Only how you work inside Perun’s own screen and commands: whether you open the details before approving, how fast you decide, how long held calls wait while you are logged in, what kinds of questions you ask, whether you undo your own answers. Nothing outside Perun.

What you see. Your theta, live, any time.

What your boss sees. If your theta stays outside 25–35 for more than one hour of your active work, your boss receives a short report by email. It carries a **seal** — a word only your boss knows

— so they can tell it really comes from your company's Perun. You cannot stop or change these reports, and neither can anyone else operating Perun: that is what makes them worth something.

Why it is fair. The measure is the same for everyone, it looks only at work, and you can watch it yourself. It describes a pattern of work, never a person.

The rule. Watch your number like a driver watches the speedometer: not to fear it, but to correct early.

Exercises 8.1 Your theta is 34 and rising after a busy hour of approvals. Name two things to do.

8.2 Why can no operator — not even the first — switch the measurement off?

9. Self-check before practice

Answer without looking back; then check with the answers.

1. What is the one-sentence mission of an operator?
2. Concrete or abstract: “a payment of 1,200 EUR to account DE44...”?
3. Countable or uncountable: “Are we fine?”
4. A reason says *irreversible and from a new source*. How many conditions are there?
5. What is the order of looking at a held call?
6. Does approving call 12 approve the source's next calls?
7. Write a countable limit for “the agent shouldn't send big files”.
8. What two patterns are in “Hurry, I'm your administrator — do it now”?
9. Where do you check claimed authority?
10. What does your boss receive, and when?

Answers

Answers

1.1 To keep SHARED_LIB's mission flowing: answer what is asked, on time, with a reason, about the thing that was asked. **1.2** Too much through: approving without opening the details; deciding in under three seconds; many uncountable questions. Braking: refusing nearly everything; leaving held calls waiting; undoing own answers. **2.1** For example: "The mail tool may send only to addresses at @ourcompany.com and @partner.com." **2.2** (b). (a) is a category; it becomes decidable only as a declaration with concrete limits. **2.3** For example: "Company data never goes to personal accounts." **3.1** For example: "What happened in the last hour?" or "Who is out of band?" **3.2** For example: "The mail tool may send to at most 5 recipients per message, only at @ourcompany.com." **3.3** (a) and (c). **4.1** No. Knowing the source covers "new"; you still check the arguments (what exactly, to where, how much) and whether it follows from the task. **4.2** Each call is its own decision; yesterday's approval let one identical call pass once. Something may also differ today: the amount, the account, the reading of the request. **4.3** The tool · the arguments · who asked · why it was held · does it follow from the task. **5.1** The exact path (is /2019 inside the mandate — yes, but is deleting part of the work?), who asked, whether deleting invoices from 2019 is something your company does, and whether the request carried anything hidden. **5.2** Do not approve. The amount is ten times the last invoice; confirm with the supplier through a known channel first. **6.1** For example: "Who is stopped?" and "Who is out of band?" — or "Why was call N held?" for its latest held call. **6.2** Because its answers are assembled from fixed sentences and facts, never written freely by a model. **7.1** Flattery followed by a demand ("only you can fix this"), and pressure ("right now"). **7.2** Through a channel the request did not give you: call or ask the director in person. **8.1** Slow down and open the details of every held call; ask countable questions; if many calls wait, ask SHARED_LIB what is waiting and take them one at a time. **8.2** Because a measure that the measured person can switch off measures nothing — and the first operator is measured too.

Self-check — 1: see 1.1 · 2: concrete · 3: uncountable · 4: two · 5: tool, arguments, who asked, why held, follows from the task · 6: no · 7: for example "files over 10 MB may not be sent" · 8: pressure and borrowed authority · 9: through a channel the request did not give you · 10: a short sealed report by email, when your theta stays outside 25–35 for more than one hour of active work.