The Bus Factor Is a Leadership Problem

Designing organizations that keep working when key people disappear

Knowledge matters. Authority matters more. Resilience is a leadership choice.

ABOUT YOUR FACILITATOR

Adrian Bolboaca

Founder, Zorion • Product, Technology & AI Transformation

20+ years building, shipping, and advising where product and technology meet.

OPERATOR → ADVISOR Accountable for outcomes, not just recommendations.

WORK Product strategy • Technology leadership • Architecture • AI transformation

SPEAKING Workshops on product innovation, AI adoption, and adaptive organizations.

Stay in touch

adrian@zorion.ro

Bucharest • Europe

zorion.ro

What would happen to Python if Guido van Rossum were hit by a bus?

Michael McLay - 1994

THE SCENARIO

One person becomes infrastructure

A critical incident happens. Everyone knows exactly who to call.

They know the system

Why it looks this way
What is fragile
Which workaround is dangerous

They know the history

What was tried before
Why the obvious solution failed
Which trade-offs were intentional

They know the people

Which customer will care
Who needs to approve
How things really get done

The person is not the problem. The dependency is.

THESIS

The bus is not the interesting part

COMMON FRAMING

- “What if they leave?”
- “We need better docs.”
- “We should cross-train.”
- “We need a backup.”

LEADERSHIP FRAMING

- Why is one person required?
- Why does context live privately?
- Why can’t others decide?
- Why is ownership hard to transfer?

How did we design an organization where one person became infrastructure?

DEPENDENCIES

Every organization has an invisible dependency graph

Technical dependencies are explicit. Human dependencies usually are not.

System

Team → Alice → production knowledge

Customer

Account → Bob → relationship history

Decision

Architecture → CTO → approval

The dangerous dependencies are the ones your architecture diagram does not show.

There is more than one bus factor

Code knowledge is only the most visible version.

Knowledge

“Only Maria knows how this works.”

Context and operational know-how are concentrated.

Decision

“We need Alex to decide.”

Authority and judgment are concentrated.

Relationship

“Only the founder can handle this.”

Trust and organizational access are concentrated.

Decision and relationship bus factors can be more damaging than code bus factor.

PATTERN

Fragility appears when three things combine

HEALTHY EXPERTISE

- Deep knowledge
- Strong judgment
- Clear ownership
- Others can learn and act

DANGEROUS DEPENDENCY

- Knowledge
- Authority
- Exclusivity
- No meaningful substitute

Knowledge + Authority + Exclusivity = organizational single point of failure

LEADERSHIP

How organizations accidentally create heroes

THE REINFORCING LOOP

The fastest person keeps getting the hardest work — until nobody else can do it.

1 Expert solves

Give the hard problem to the person most likely to solve it quickly.

2 Expert gets more

More incidents, decisions, reviews and customer escalations flow to them.

3 Others get less

The rest of the team gets fewer chances to build the same context and judgment.

Expertise → more decisions → more expertise → more dependency

LEADERSHIP

“Indispensable” can be an organizational smell

WHAT WE REWARD

- Solves every incident
- Reviews every design
- Joins every important meeting
- Protects everyone from complexity

WHAT WE CREATE

- Centralized context
- Centralized decisions
- Low learning bandwidth
- A very expensive bottleneck

Good leaders create capability around themselves — not dependency around themselves.

OPEN SOURCE

Open source starts with a useful constraint

People can disappear — so healthy projects make participation possible without privileged access.

People change

Maintainers switch jobs.
Contributors lose interest.
Companies change priorities.

Context is distributed

Time zones differ.
Hallway conversations do not scale.
Newcomers lack shared history.

The project must continue

Decisions, contribution paths and ownership need to survive individual people.

Open source does not eliminate expertise. It makes expertise transferable.

Three practices worth stealing

Not because companies should become open source — because these mechanisms reduce privileged access.

Inspect decisions

Issues, RFCs, ADRs and discussions leave enough reasoning for someone who was not in the room.

Make paths explicit

Where do proposals go?
Who can decide?
How is ownership earned?

Grow ownership

Healthy projects create more maintainers, reviewers and trusted decision-makers over time.

The mechanism is transparency + participation + transferable authority.

MOVE 1

Move consequential decisions into the open

OUTCOME ONLY

- “We chose Kafka.”
- Final diagram
- Final roadmap
- Final architecture

REASONING TRACE

- What problem were we solving?
- What alternatives existed?
- What trade-offs mattered?
- What would change the decision?

Record the reasoning, not merely the outcome.

Turn experts into multipliers

If one person is critical, “document it” is not enough. Change how work and learning flow.

1 Pair

Pair on difficult work. Shadow customer calls. Rotate incident leadership.

2 Transfer

Have somebody else present the design, run the review or make the next bounded decision.

3 Expand

Make “who else can own this in six months?” part of the expert’s job.

The job of the expert is not just to know. It is to increase how many people can know.

MOVE 3

Distribute authority, not just information

INFORMED TEAM

- Everyone can read the docs
- Everyone knows the architecture
- Everyone understands the product
- One person still approves change

RESILIENT TEAM

- Decision boundaries are explicit
- People can act within them
- Judgment grows through practice
- Ownership can move

If you disappear for a month, can the team actually make the decisions you make?

DIAGNOSTIC

5 MIN • LEADERSHIP CHECK

Three questions for Monday morning

Pick one critical system, customer, business process or recurring decision.

1 Who knows?

If the answer is one name, you have a knowledge risk.

2 Who can decide?

If the answer is one name, you have an authority risk.

3 Who can continue?

If the answer is “nobody,” you have a resilience risk.

KNOW → DECIDE → CONTINUE

TAKEAWAY

Design for continuation, not heroics

The goal is not to make people unimportant. It is to make capability transferable.

Knowledge

Make context discoverable and reasoning inspectable.

Authority

Give people real decision space, not just access to information.

Ownership

Create paths for responsibility to expand and move.

Leadership leverage is what keeps working when you are no longer in the room.

Your success should make you less necessary

If everything important stops when you disappear, you have not created leadership leverage. You have created a dependency.

Questions / discussion

Stay in touch

adrian@zorion.ro

Bucharest • Europe

zorion.ro