Designing for Failure: Resilience and Antifragility in the Management of Critical Projects [1]

Luca Paolo Giuseppe Prinzio

Abstract

In critical systems built on cloud infrastructure and artificial intelligence, the unforeseen is not an exception to be eliminated but a structural condition. Technology evolves faster than the culture, governance and operational capacity that ought to govern it: between what a project plans and what the system actually does there remains an irreducible margin, and it is within that margin that incidents emerge. This article argues that, in such contexts, the objective of project management can no longer be limited to avoiding failure: it must also be to design the project-system so that it withstands and recovers when an unforeseen failure occurs, and so that the organisation learns from it.

The article distinguishes between robustness, resilience and antifragility, and applies the distinction to project practice while explicitly recognising its limits: in critical services and in Public Administration the realistic objective is resilience, whereas antifragility properly concerns processes, teams and learning, not the delivered service. It identifies the dimensions that determine whether a project withstands the shock — team cohesion, human sensors, the map of breaking points, decision-making under stress — and proposes operational practices drawn from reliability engineering: chaos engineering applied to management, cognitive redundancy, the error budget, the blameless postmortem. It closes by discussing the ambivalent role of artificial intelligence, which widens the gap between technology and culture and is itself a source of the unforeseen, but which can also strengthen the early detection of anomalies.

Keywords: *project resilience, antifragility, chaos engineering, cognitive redundancy, error budget, Safety-I and Safety-II, high reliability organisations, requisite variety, critical systems, cloud, artificial intelligence, Public Administration*.

[1] How to cite this article: Prinzio, L. P. G. (2026). Designing for Failure: Resilience and Antifragility in the Management of Critical Projects, featured paper, *PM World Journal*, Vol. XV, Issue X, October 2026

© 2026 Luca Paolo Giuseppe Prinzio

1. Introduction: The Unforeseen as a Structural Condition

A critical digital service, hosted on cloud infrastructure, had been carefully designed. There was a project plan; the tests had been passed; runbooks, a governance model and continuity procedures were all in place. Each component, taken individually, had been verified. Then, one night, the service degraded in a way no one had foreseen: an automatic update to a library, combined with an anomalous traffic peak and the silent degradation of an external dependency, produced a behaviour that no document contemplated. None of those three events, in isolation, would have been a problem. Their combination was. The recovery, in the end, did not depend on the quality of the plan — the plan did not foresee that scenario — but on the ability of a few people to improvise well, under pressure, reconstructing on the fly what was happening.

This scene is not an exceptional incident. It is the ordinary way in which complex systems fail. Many traditional plan-and-control interpretations of project management implicitly assume that, with sufficient planning, risk analysis and control, failure can be prevented. Contemporary standards are considerably more nuanced — the seventh edition of the PMBOK® Guide devotes a performance domain to uncertainty, and agile approaches are built around adaptation — but the assumption survives in organisational practice, where governance is still frequently designed as though the future were enumerable. In critical systems built on cloud infrastructure and artificial intelligence, that assumption does not hold. Not because planning is useless, but because there is a class of events that no plan can anticipate: unforeseen interactions between components, under conditions that no one has specified, at a speed that no control can match.

The argument of this article is that, in these contexts, the objective of project management must shift. **No longer merely to avoid failure — which no amount of planning can guarantee — but also to design the project-system so that it withstands and recovers when an unforeseen failure occurs, and so that the organisation emerges from it better prepared.** It is the move from a defensive conception, in which safety coincides with the absence of faults, to an adaptive conception, in which safety is the capacity to keep functioning under changing conditions — the distinction Hollnagel formulates as that between Safety-I and Safety-II. The analysis focuses on high-criticality contexts — infrastructure, ICT, Public Administration — where the cost of failure is high and where, precisely for that reason, the temptation to believe that risk can be reduced to zero is stronger and more dangerous.

© 2026 Luca Paolo Giuseppe Prinzio

2. Methodological Approach

This article does not present the results of a single piece of original empirical research. It is a practice-based reflection, combining a secondary analysis of technical and organisational literature with the direct observation of recurring patterns in complex project contexts: critical infrastructures, cloud platforms, cybersecurity, operational resilience, and digital transformation in Public Administration and large organisations.

The frame of reference draws on domains that project management literature rarely visits. From reliability engineering and Site Reliability Engineering come the concepts of chaos engineering, error budget and blameless postmortem. From the study of high reliability organisations (Weick and Sutcliffe) come preoccupation with failure and deference to expertise. From resilience engineering (Hollnagel; Hollnagel, Woods and Leveson) comes the distinction between safety understood as the absence of faults and safety understood as adaptive capacity. Perrow's analysis of accidents in high-risk systems, Taleb's work on fragility and antifragility, Ashby's principle of requisite variety, and Wucker's distinction between sudden and announced risks complete the picture. The cases presented are anonymised and generalised: they do not refer to specific organisations, projects, suppliers or persons, but describe situations observed with sufficient frequency to be treated as recurring patterns.

3. The Irreducible Residue: Why the Gap Does Not Close in Cloud and AI Systems

Every project lives with a distance between what is planned and what happens. In traditional systems this distance can, with discipline and control, be reduced until it becomes manageable. In critical systems built on cloud infrastructure and artificial intelligence, however, part of that distance is irreducible. It is worth understanding why, because everything else follows from this understanding.

The first reason is speed. Technology evolves faster than the culture and the operational capacity that ought to govern it. A cloud capability or an AI model can be acquired overnight; the organisational maturity to manage them — the competencies, the exercised procedures, the shared mental models — takes months or years. Incidents emerge in that temporal gap: between what the system can already do and what the organisation is not yet equipped to govern.

© 2026 Luca Paolo Giuseppe Prinzio

The second reason is structural. Perrow showed that in systems characterised by interactive complexity and tight coupling, system accidents are not anomalies but expected outcomes: when many components interact in non-linear ways and a fault propagates rapidly before anyone can intervene, occasional failure is a property of the system, not a mistake of those who run it. Modern cloud architectures — with their dynamic dependencies, pervasive automation and third-party services — are a textbook instance of this condition.

The third reason is epistemic: it is not possible to specify in advance all the states of a sufficiently complex system. Resilience engineering formalises this point by distinguishing two conceptions of safety. Safety-I understands safety as the absence of faults: one identifies the ways in which the system can break and builds barriers to prevent them. This is useful, but it presupposes that the ways of failing can be enumerated in advance. Safety-II understands safety as the capacity to function well even under variable and unexpected conditions: it does not eliminate variability but cultivates the system's ability to adapt to it. The first prevents what can be imagined; the second prepares for what cannot. In cloud and AI systems, the second is not an optional addition: it is the part of the problem that the first cannot cover.

The practical consequence is clear. If a share of the unforeseen is structural, then a project that stakes everything on prevention is optimising only half of the problem. The other half — what happens when the unforeseen occurs — requires a different property, one that is not built with more controls, but with more capacity to absorb, react and learn.

4. Robust, Resilient, Antifragile: An Honest Taxonomy

To reason precisely, one must distinguish properties that are often confused. Building on Taleb's distinction between fragility, robustness and antifragility, and on the vocabulary of resilience engineering, four different responses to stress can be separated out. A fragile system breaks under stress. A robust system resists stress without changing: it takes the blow and remains as it was. A resilient system suffers the blow but recovers, returning to service and absorbing the disturbance. An antifragile system, finally, improves through stress: disorder, within limits, makes it stronger. Taleb himself treats robustness and resilience as close neighbours; the distinction between them is drawn here because, in project practice, resisting a shock unchanged and recovering from it are different capabilities and require different investments.

© 2026 Luca Paolo Giuseppe Prinzio

Applying this taxonomy to projects requires honesty, because the concept of antifragility is nowadays overused. In critical services and in Public Administration, the realistic and legitimate objective is resilience, not antifragility. One cannot deliberately fail an essential service to citizens — healthcare, the civil registry, payments — in order to "learn" or to "grow stronger". The level of service delivered must be protected, not exposed to disorder. Antifragility, in these contexts, does not apply to the service, but to processes, teams and organisational learning: it is these that can and must improve with every incident, while the service itself remains protected. Confusing the two planes produces a dangerous rhetoric that mistakes irresponsible exposure to risk for managerial courage.

A second distinction guards against the same abuse. Not everything we call a "black swan" truly is one. Taleb reserves the term for genuinely unpredictable events, rare and comprehensible only in hindsight. Wucker observes that most organisational disasters belong to a different category: the **grey rhino**, the highly probable and clearly visible danger that one chooses to ignore. The critical dependency everyone knew was fragile, expertise concentrated in a single person, the disaster recovery plan never exercised: these are not black swans, they are grey rhinos. Calling them unpredictable, after they have run us over, is an alibi. Authentic resilience begins with the honesty to distinguish the few events that could not have been foreseen from the many that could, but that no one was willing to look at.

5. The Dimensions of Project Resilience

The resilience of a project is not a single attribute that one either possesses or lacks. It is the resultant of several distinct properties, each of which determines how the project-system will react when the shock arrives. Four in particular deserve attention, because they act in the moments when the plan is no longer enough.

Team cohesion. Under stress, the difference between a project that holds and one that collapses is often the quality of internal relationships. A cohesive group absorbs the shock: mutual trust allows coordinated improvisation, rapid communication, the fluid redistribution of tasks in the moment of crisis. A corroded group, by contrast, amplifies the shock. Felps and colleagues documented how even a single dysfunctional behaviour can depress the performance of an entire group (for an application to project management, see Prinzio, 2026): under pressure, that relational fragility becomes the first point of rupture. Cohesion is not a "soft" value: it is what holds the team together when procedures are not enough.

© 2026 Luca Paolo Giuseppe Prinzio

Human sensors. Resilience depends on the capacity to detect the anomaly early, while it is still small and correctable. The most sensitive sensors of a project are people: those who sense that something is off long before the data confirms it. But these sensors only work if people feel free to raise doubts, mistakes and bad news without fear of punishment — what Edmondson calls psychological safety. High reliability organisations deliberately cultivate a preoccupation with failure: they treat every small signal as information, not as noise. Where bad news is punished, the sensors switch off, and the system loses its sight precisely when it needs it most.

The map of breaking points. Not all parts of a system give way in the same manner. A resilient project knows its critical dependencies and the blast radius of each possible fault: it knows which components are tightly coupled — where a local fault propagates fast — and which are isolated. The most dangerous dependencies are those taken for granted: the network, the supplier, the external service that "will always be there", and that no one has ever put to the test. An unmanaged dependency is a risk disguised as an assumption. Mapping the breaking points means knowing in advance where the system will give way, and therefore where to concentrate defences and containment.

Decision-making under stress. When the plan breaks down, the speed and quality of the decision matter more than its orthodoxy. In those moments, decision authority cannot remain rigidly at the top: it must be able to migrate towards whoever holds situational knowledge — the engineer watching the logs, the operator who knows the system — and then return once the crisis has passed. Weick and Sutcliffe call this principle deference to expertise: during the emergency, the person who knows commands, not the person with the title. A governance model that cannot temporarily cede control to whoever is closest to the problem turns every incident into a crisis of slowness.

Dimension	What it protects	What weakens it
Team cohesion	Absorbs the shock: cohesion enables coordinated improvisation	Corrosive behaviours, distrust, emotional overload
Human sensors	Detects the anomaly early: people flag what is off	Blame culture, silence, absence of psychological safety
Map of breaking points	Contains the damage: one knows which dependencies give way, and how far	Dependencies taken for granted, unmanaged tight coupling
Decision-making under stress	Reacts in time: authority migrates to whoever knows, then returns	Rigid governance, centralised decisions, late escalation

© 2026 Luca Paolo Giuseppe Prinzio

These four dimensions are not built with more documents. They are built with practices that deliberately put the system to the test, before reality does. It is to these practices that the following sections are devoted.

6. Chaos Engineering Applied to Management

In the world of distributed systems, an apparently paradoxical practice has emerged: deliberately injecting faults into a running system in order to verify, empirically, that it holds. Instead of assuming that the architecture is resilient, one puts it to the test by introducing controlled failures into parts of it and observing how it reacts. The discipline favours experiments conducted as close to production as possible, since that is where real behaviour appears, but in practice it is applied along a graduated scale of exposure — from test and production-like environments to live systems — with the blast radius deliberately contained at each step. Chaos engineering arises from a simple idea: confidence in the resilience of a system should rest not on hope, but on evidence.

This logic can be transposed to management. It is not enough to write continuity plans and emergency procedures: they must be exercised under realistic conditions, and one must observe where they give way. ***Game days*** and *tabletop* exercises do exactly this: they simulate an adversity and observe how the human system reacts. What happens if the sponsor suddenly becomes unreachable at the moment of a critical decision? If a key person is absent without notice during a delicate phase? If an essential supplier does not respond? Simulating these events — not discussing them in the abstract, but staging them — reveals how the organisation actually reacts, rather than how it believes it would react.

A particularly instructive variant is the **documentation test**: handing a runbook to someone who did not write it and asking them to carry out the procedure. Too often documentation is comprehensible only to its author, and becomes useless precisely when it is needed — when the author is on holiday, has changed role or is otherwise unavailable. Verifying that a procedure can be performed by another person is a direct way to measure the *bus factor* of the project: how many people would have to be unavailable, or leave the organisation, for a critical capability to become unmanageable.

It is important to distinguish this practice from the *premortem*. The premortem — imagining, before starting, that the project has already failed and asking why — is a valuable instrument, but it is imaginative and ex ante: it interrogates our assumptions. Chaos engineering is empirical: it does not ask what might break, it breaks something to see what actually happens. The first explores the map; the second walks the terrain. They are

© 2026 Luca Paolo Giuseppe Prinzio

complementary but not interchangeable: many fragilities become visible only when the fault is real rather than hypothesised.

There is a theoretical reason why these tests are indispensable precisely in cloud and AI systems. Ashby formulated the principle of *requisite variety*: to govern a system, a controller must possess a variety of responses at least equal to the variety of states the system can assume. When the complexity of a system expands — as it does with the pervasive automation and dynamic behaviours of cloud platforms and artificial intelligence — a rigid control system, with a fixed repertoire of responses, is inevitably outmatched. Injecting stress in a controlled way serves precisely to measure this: whether the control system has sufficient variety, or whether it collapses when confronted with a situation for which it has no response in its repertoire. And where rigid control is not enough, the only alternative is adaptive control: procedures that admit conscious deviation, and authority that can move towards whoever, at that moment, holds the required variety of responses.

Case 1 – The runbook no one could execute. An organisation had a formally complete recovery plan for a critical service. During a game day, the procedure was handed not to the expert who had written it, but to the on-call engineer who would realistically have had to run it at three in the morning. The documentation, which on paper appeared exhaustive, proved impossible to follow: it took for granted steps that only the author knew, relied on implicit knowledge that had never been formalised, and some credentials had changed without the document being updated. The plan existed, but the procedure could not be executed by the person expected to use it. None of these problems was technical: they were organisational fragilities made visible only by putting the procedure to the test instead of taking it for granted.

7. Cognitive Redundancy and Slack

If a share of failure is inevitable, the capacity to absorb it depends on the reserves the system has set aside. Resilience, in this sense, has a cost: it requires redundancy, and redundancy looks like inefficiency — until it is needed.

The most critical form of redundancy in projects is not technical but cognitive. It means, first, ensuring that no critical knowledge resides in a single person: a bus factor greater than one for every critical role and capability. But it also means something subtler than shared documents. A resilient team shares mental models, not only procedures: it can reconstruct the intent behind a choice — why something was done in a particular way — and not merely the sequence of steps. It is this understanding of intent that allows a group to improvise

© 2026 Luca Paolo Giuseppe Prinzio

correctly when the situation moves beyond the manual, because no manual covers every case.

The second form of reserve is *slack*: a deliberate margin of capacity — of time, people, attention — not allocated to any task, kept free to absorb the unforeseen. Slack is not waste: it is insurance. A system used at one hundred per cent of its capacity has no margin with which to absorb a surprise; the first unexpected event pushes it into saturation. This is another face of requisite variety: redundancy is what gives the system the variety of responses with which to react to novel situations.

Here an uncomfortable truth must be stated: efficiency and resilience are in tension. Every margin cut in the name of optimisation is a margin taken from the capacity to absorb a shock. Misapplied lean practices — lean reduced to cost cutting, rather than the discipline that distinguishes waste from protective capacity and manages variability — together with cost-containment pressure and certain public procurement constraints, systematically push towards eliminating the very redundancy that resilience requires. This is not an argument against efficiency, but against efficiency pursued blindly: whoever governs critical projects must be able to defend slack as a conscious choice, rather than have it cut away as waste. Eliminated redundancy is invisible while all goes well; it makes itself felt, with interest, on the day something gives way.

Case 2 – Knowledge in a single head. In an infrastructure project, full command of a critical component was concentrated in a single specialist. Everything worked, and that person's efficiency was in fact celebrated: they solved every problem quickly, alone. During a real incident, however, they were unreachable. The team, lacking shared mental models of that component, could not intervene with confidence; recovery took far longer than necessary, not because of a technical limit, but because of the absence of cognitive redundancy. The efficiency that had been rewarded proved, in retrospect, to be the cause of the fragility.

8. Artificial Intelligence: Accelerator of the Gap and Instrument of Resilience

Artificial intelligence enters this discussion with two faces, and ignoring either of them leads to mirror-image errors.

On the one hand, AI is an accelerator of the gap. It accelerates the technical layer — speeding up development, automation and scale — while the human and cultural layer

© 2026 Luca Paolo Giuseppe Prinzio

remains as slow as ever. The result is a widening gap between what the system can do and what the organisation knows how to govern — the very gap in which incidents emerge. Moreover, AI introduces a new source of the unforeseen: its behaviours can be emergent and non-deterministic, its dependencies opaque. A model that behaves impeccably in testing may react differently when exposed to real-world data, in ways no one has specified. This is exactly the kind of interactive complexity that makes system failure structural.

On the other hand, AI is an instrument of resilience. It can strengthen observability and the early detection of anomalies, acting as an additional sensor that catches deviations before they become incidents. Here a principle matured in Site Reliability Engineering is useful: the *error budget*. Treating one hundred per cent reliability as an absolute objective is usually misguided — unattainable in practice and extremely costly. Instead one defines a share of tolerated failure: as long as one remains within that budget, one can innovate rapidly; when it is exceeded, one stops and stabilises. It is a cultural reversal of the zero-defect obsession, and it applies to management as well: declaring how much imperfection is acceptable turns failure from a taboo into a governable variable.

One risk remains to be named: automation that conceals fragility. "If the AI has not alerted me, everything is fine" is a false inference. Algorithmic silence is no guarantee of health: a system that appears under control because no alarm has sounded may be fragile precisely where the monitoring does not look. Artificial intelligence widens the capacity to see what is in the data; it does not, on its own, widen the capacity to withstand what is not yet in the data. Resilience remains a property of the socio-technical system as a whole, not a service one can purchase.

9. Designing for Failure: What to Do Operationally

Designing for failure means shifting part of one's attention from preventing failures to containing and recovering from their consequences. Some practices, drawn from reliability engineering and adapted to management, make this shift concrete.

Reduce the blast radius. Design the system — technical and organisational — so that a local fault stays local. Decouple components, isolate critical functions, contain propagation: a problem in one area must not drag down the whole project. It is the direct antidote to tight coupling.

Provide for graceful degradation. A resilient system, when something gives way, does not collapse entirely: it relinquishes non-essential functions and preserves the critical ones.

© 2026 Luca Paolo Giuseppe Prinzio

The same principle applies to project plans: a plan that degrades in a controlled way — one that establishes in advance what may be sacrificed in order to protect the essential — is more robust than a plan that works only if everything goes as expected.

Declare an error budget. Make explicit how much failure is acceptable, and use that threshold as a lever of governance: within the budget one innovates; beyond it one stops and consolidates.

Adopt the blameless postmortem. After every incident, reconstruct precisely what happened in order to learn, not to attribute individual blame. The search for a culprit destroys the human sensors: if raising a problem exposes one to punishment, problems stop being raised. It is here that antifragility is legitimate and indeed a duty — not in the service, but in learning: every incident should leave the organisation more capable than before.

Exercise, do not only document. Periodic game days and tabletop exercises; readiness verified continuously rather than ticked once; testing the documentation with someone who did not write it. Readiness is a state that decays: it must be re-measured, not certified once and for all.

Ensure minimum redundancy. Explicit bus-factor policies, shadowing, knowledge rotation, and slack defended as a choice rather than suffered as waste.

A real constraint must be acknowledged. In highly regulated contexts, such as Public Administration, injecting faults into production is often impracticable or irresponsible: one does not experiment on a service that is essential to citizens. In these cases one favours simulated exercises, test and production-like environments, tabletop sessions and a progressive approach that widens exposure only as confidence is earned. The principle does not change — to put the system to the test before reality does — but its application adapts to the level of tolerable risk. Antifragility, once again, is cultivated in processes and learning, while the service itself remains protected.

10. Conclusion

In critical systems built on cloud infrastructure and artificial intelligence, the unforeseen is not a defect to be corrected: it is a condition to be lived with. Technology runs faster than the culture that ought to govern it, interactive complexity makes system failure structural, and no specification can enumerate in advance all the ways in which things will turn out

© 2026 Luca Paolo Giuseppe Prinzio

differently from what was planned. Faced with this reality, the pursuit of perfect control is not merely unrealistic: it is dangerous, because it diverts energy from the capacities that truly matter.

Two such capacities should be kept distinct, because conflating them is the most common error in this field. **The first is resilience: to withstand the shock and recover — to keep the critical service standing and return it to normal operation.** The second is antifragility, and it belongs elsewhere: it is the capacity to turn each ordeal into an improvement of processes, teams and organisational learning. The delivered service must be protected, not exposed to disorder in order to strengthen it; what must grow stronger with every incident is the organisation around it. Both are built with concrete practices: putting the system to the test before reality does, keeping cognitive reserves and margins of capacity, containing the reach of faults, and learning from every incident without seeking culprits.

In the end, however, a resilient system is made of people. **No architecture, no runbook and no algorithm holds a project together at the moment the plan breaks: it is held together by people who can improvise well as a group, because they trust one another and understand the common intent.** Resilience does not arise from the perfection of the plan, but from the cohesion and the readiness of those who carry it out. And so the ultimate criterion by which to judge a project in these contexts is not whether it avoided failure altogether — an outcome no one can guarantee — but how well it stays on its feet when failure arrives, and what the organisation learns once it has passed.

References

Ashby, W. R. (1956). An introduction to cybernetics. Chapman & Hall.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.). (2016). Site reliability engineering: How Google runs production systems. O'Reilly Media.

Edmondson, A. C. (2018). The fearless organization: Creating psychological safety in the workplace for learning, innovation, and growth. Wiley.

Felps, W., Mitchell, T. R., & Byington, E. (2006). How, when, and why bad apples spoil the barrel: Negative group members and dysfunctional groups. Research in Organizational Behavior, 27, 175–222.

Hollnagel, E. (2014). Safety-I and Safety-II: The past and future of safety management. Ashgate.

Hollnagel, E., Woods, D. D., & Leveson, N. (Eds.). (2006). Resilience engineering: Concepts and precepts. Ashgate.

Klein, G. (2007). Performing a project premortem. Harvard Business Review, 85(9), 18–19.

© 2026 Luca Paolo Giuseppe Prinzio

Perrow, C. (1984). Normal accidents: Living with high-risk technologies. Basic Books.

Prinzio, L. P. G. (2026). The team performance paradox: How a single toxic behavior can sabotage a project. PM World Journal, XV(V). https://pmworldlibrary.net/wp-content/uploads/2026/05/pmwj164-May2026-Prinzio-The-Team-Performance-Paradox.pdf

Project Management Institute. (2021). A guide to the project management body of knowledge (PMBOK® Guide) (7th ed.) and The standard for project management. Project Management Institute.

Raisch, S., & Krakowski, S. (2021). Artificial intelligence and management: The automation–augmentation paradox. Academy of Management Review, 46(1), 192–210.

Rosenthal, C., & Jones, N. (2020). Chaos engineering: System resiliency in practice. O'Reilly Media.

Taleb, N. N. (2007). The black swan: The impact of the highly improbable. Random House.

Taleb, N. N. (2012). Antifragile: Things that gain from disorder. Random House.

Weick, K. E., & Sutcliffe, K. M. (2007). Managing the unexpected: Resilient performance in an age of uncertainty (2nd ed.). Jossey-Bass.

Wucker, M. (2016). The gray rhino: How to recognize and act on the obvious dangers we ignore. St. Martin's Press.

Note on the Use of AI

During the preparation of this article, artificial intelligence tools were used to assist in translation from Italian to English, in linguistic revision and in the organisation of the text. All substantive content, analyses, arguments and conclusions are the work of the author, who retains full responsibility for the originality, accuracy and validity of the work presented.

Note on Anonymisation

The cases described in this article are anonymised and generalised. They do not refer to specific organisations, projects, suppliers or persons. They are situations observable in many complex projects and have been formulated generically to allow their analysis without reference to identifiable contexts.

© 2026 Luca Paolo Giuseppe Prinzio

About the Author



Luca Paolo Giuseppe Prinzio

Turin, Italy

Luca Paolo Giuseppe Prinzio is a certified Project Manager and Database Administrator at CSI Piemonte in Turin, Italy, where he participates in complex projects on cloud and security. For over twenty years he has worked in the ICT world and carries out teaching and consulting activities in the field of Project Management. He can be contacted at lprinzio@gmail.com and linkedin.com/in/lprinzio

© 2026 Luca Paolo Giuseppe Prinzio