Mistakes To Learn From

Enrique Matta-Rodriguez

Amberton University

HRT6570.E1

Dr. John Sinclair

Feb 21, 2026

Scenario

In 2015, I served as the unofficial lead for the Global Sales and Contract Engineering QA team at Rackspace. As the most senior person in the domain, I was responsible for a diverse group of seven, both full-time employees and contractors from the U.S., Russia, and India. We were a highly capable team of engineers who wrote our own automation and handled complex code, yet we were still in the fragile Forming stage, having worked together for only a few weeks.

The breaking point came when a major release failed, triggering a significant production outage. We had missed critical testing requirements, and the pressure was immense. During the postmortem, I failed the most basic test of leadership. Instead of facilitating the blameless retrospective common in the tech industry, I lost my composure. I took the team into a room and castigated them harshly, raising my voice in an unprofessional outburst that erased any progress we had made.

That moment showed me that effective leadership is only sustainable when you have built sufficient trust and equity with your team. By choosing to vent my frustration rather than provide a path forward, I shattered the psychological safety required for high-level engineering. Trust is notoriously difficult to earn and incredibly easy to lose. My actions trapped the team in the Storming phase, and it took significant time even to begin to understand the depth of the damage I had done to our culture.

Analysis of Team Formation

Daft outlines five stages of team development: Forming, Storming, Norming, Performing, and Adjourning. By yelling at the team during a crisis, I effectively locked us into the Storming

phase. In this stage, strong personalities emerge, and conflict over goals or methods intensifies (Stein). While such conflict can be productive, Daft emphasizes that it must be carefully managed. My angry reaction, instead of a blameless approach, reinforced the power struggles of Storming. Rather than progressing into Norming, where differences are resolved, and cohesion develops, the team likely slipped into defensiveness and distrust (Daft; Stein).

This experience also underscored the importance of selecting the right communication channel. Our in-room meeting was rich in cues because everyone was physically present, but the emotional climate turned destructive. My frustration and raised voice created a negative tone, setting off an emotional contagion that undermined problem-solving. In contrast, an effective retrospective should shift from a centralized to a decentralized communication network, encouraging open and honest sharing. By castigating the team, I unintentionally shut down upward communication; fear of my reaction made people less likely to surface bad news or missed requirements.

Empowerment requires giving team members the authority, information, and rewards they need to make decisions, with the goal of team self-direction (Daft). Daft notes that empowerment rests on trust. By blaming a diverse group, including vulnerable contractors, for the production outage, I effectively de-powered us. Such harsh criticism does not build the commitment required for high-quality QA; it promotes mere compliance rather than genuine engagement.

I distinguish between position power (rank-based) and personal power (grounded in expertise and relationships) (Daft). As the most senior and knowledgeable member of the team, I initially relied on personal power. However, during the outage, I shifted from expertise-based influence to coercive power by raising my voice and castigating the team (Daft). As Daft notes,

coercive power often provokes resistance or passive compliance. In a detail-oriented QA environment, a fear-driven team is less likely to catch subtle bugs or speak up about concerns.

To repair this, I need to create a communication climate that promotes upward information flow. Daft emphasizes that high-performing teams require a psychologically safe environment for discussing errors. By replacing a blameless retrospective with a reprimand, I undermined that safety and signaled to the two employees and five contractors that mistakes lead to punishment, not analysis. This dynamic encourages information hoarding and blame shifting, which block progress into the Performing stage (Daft; Stein).

Our diverse team (Russia, India, U.S.) was already struggling in the Storming phase. The harsh, low-context communication style I used likely intensified cultural differences around communication and authority, alienating some members and widening the gap between contractors and staff. This aggressive approach prevented us from building the social cohesion needed to move into the Norming phase (Daft; Stein).

Under the Hersey and Blanchard Situational Leadership Model, a team in Storming typically needs a high-task/high-relationship Coaching style (Daft). Instead, I defaulted to a hostile high-task/low-relationship Telling style (Daft). Given the team's Forming/Storming level of development, they needed me to coach them through the outage (Daft; Stein). My emotional reaction denied them the relationship support they needed to learn from the missed requirements.

Daft also notes that subconsciously categorizing team members can create an out-group, in this case, the five contractors relative to the two full-time employees. My behavior likely reinforced this divide, violating Daft's inclusive leadership requirements for both belongingness and uniqueness. Because coercive power is interpreted differently across cultures (for example, in Russia and India), my approach may have amplified feelings of alienation (Daft). As

relationships erode, morale falls, and people begin to feel like resources managed by fear rather than valued contributors.

In a technical environment like Rackspace, innovation depends on psychological safety. Daft argues that people must feel free to fail and to share unconventional ideas. By skipping the blameless retrospective, I signaled that it was not safe to openly discuss errors, an exclusionary practice that undermines the diversity of thought needed for innovation. Yelling at the team for missed requirements further punished risk-taking. Instead of investing energy in better testing frameworks, team members likely shifted to CYA behavior, abandoning creative solutions to avoid provoking my anger.

AI Analysis (From ChatGPT)

Here's how you could've handled it better, based on what you wrote about the team being new (Forming/early Storming), culturally diverse, and under outage pressure.

1) In the moment: don't do the "closed-door blast"

You took everyone into a room and vented. That's the single biggest misstep because it traded short-term emotional release for long-term fear.

What to do instead:

- Call a 10–15 minute "stabilize first" huddle: "We're in outage mode. We're going to restore service, then we'll learn."
- Separate incident response from learning: no performance talk while adrenaline is high
- If you're heated, delegate facilitation to the calmest senior engineer or your manager. Leaders don't get bonus points for being the loudest person in the room.

2) During the postmortem: run a real blameless retro (structure beats emotion)

You already know blameless is the standard. The missing piece is a *scripted* process that prevents it from turning into blame. A solid retro flow:

- Timeline first (facts only): what happened, when, signals, decisions, deployments, rollback steps.
- Contributing factors: requirements gap, coverage gaps, ownership ambiguity, time pressure, comms breakdown, tooling gaps.
- "How did our system allow this?" not "Who did this?"
- Action items with owners and dates (small number, high impact): add missing test gates, redefine release criteria, update checklists, monitoring, canary rules, etc.

3) How you should've talked to the team (what to say instead of yelling)

When a release fails, people are already stressed and embarrassed. Your job is to turn that into learning and coordination. Use language like:

- "This is serious. We're going to fix it and make sure it doesn't repeat."
- "I own the process. We missed requirements, which means our system failed."
- "I need everyone's help surfacing what went wrong without fear."

That keeps urgency without humiliation.

4) Handle the "contractor vs FTE" dynamic explicitly

You had five contractors and two employees. Contractors are automatically more vulnerable. Public pressure or even aggressive tone can instantly create silence and "do the minimum."

Better moves:

- Make it clear accountability is about the process, not employment status.
- Rotate ownership of improvements so contractors can contribute visibly.
- If performance is truly an issue, handle it 1:1, not in a group blow-up.

5) Repair move you should've made immediately after the outburst

If you mess up publicly, you fix it publicly. Waiting makes it rot. What you should've done within 24 hours:

- Short apology to the team: "I was out of line. That's on me."
- Re-commit to blameless learning: "We're redoing the retro properly."
- Ask for input safely: anonymous doc, 1:1s, or have someone else collect feedback.

That's how you rebuild psychological safety after you puncture it.

6) The "coach" posture you called out: make it real

You wrote that a Storming team needs coaching. Concretely, that means:

- Clear roles during incidents (incident commander, comms lead, ops, QA validation).
- Clear definition of "done" for testing requirements.
- Pre-release checklist and test strategy that matches risk.
- Regular cadence: weekly quality review, so the postmortem isn't the first time problems surface.

Human Analysis of ChatGPT

The AI analysis of the complex hypothetical scenario accurately diagnosed structural and process deficits but offered an emotionally tone-deaf and insufficiently nuanced reply. Despite this emotional shortcoming, it provided six actionable areas for improving team dynamics.

Though a direct application is unfeasible, the profound lessons are adaptable to my current leadership role. The core realization is the necessary synergy between leading (guiding, directing, influencing) and coaching (a thought-provoking process inspiring potential, per PMI).

My error was conflating the roles, defaulting to the "directing" aspect (a punitive, fault-finding approach) when coaching was needed (Daft). The response should shift from castigation to collaborative analysis, with the team sitting down to examine systemic failures and guide self-correction. The vital lesson is changing perspective: from judging past failure to facilitating future success and empowering the team.

Conclusion

The Rackspace incident was a stark but valuable demonstration of the importance of psychological safety and strong leadership. Although crises can push leaders toward coercive tactics, genuine leadership requires maintaining a coaching approach, especially during the “storming” phase when a team is struggling. A leader must redirect the focus from assigning individual blame to conducting a systemic analysis. This shift is essential because it turns a production failure into a catalyst for organizational improvement rather than a source of cultural breakdown. Moving forward, the priority must be to cultivate a communication environment built on trust, turning every setback into an opportunity to strengthen and empower a diverse team for sustained, long-term success.

Reference

Leading, coaching, and managing. (n.d.). Www.pmi.org. https://www.pmi.org/learning/library/leading-coaching-managing-hat-wear-6029

Daft, R. L. (2021). The leadership experience (8th ed.). Cengage Learning.

Stein, J. (2024). Using the stages of team development. Massachusetts Institute of Technology; MIT Human Resources. https://hr.mit.edu/learning-topics/teams/articles/stages-development

ChatGPT - Situation Handling Review. (2026). ChatGPT. https://chatgpt.com/share/6999cddd-eab8-800b-b62c-daad096d8168