

# Leveraging the Two-But Rule for Innovation in Technical Teams

Part 5. Strategy, AI, and Team Culture

This chapter explores the "Two-But Rule," a method for enhancing innovation within technical teams by pairing each critique with a solution-focused alternative. By transforming...

## In this chapter

1. Introduction
2. How the Two-But Rule Works
3. Addressing the Challenges of Feedback and Innovation
4. The Role of Intentions in Innovation
5. Balancing Innovation and Practicality in Technical Teams
6. Applying the Two-But Rule in Software Engineering
7. Best Practices for Fostering Innovation in Technical Teams
8. Conclusion
9. Discussion Prompts

## 🕒 Chapter overview

**Key Topics:** Structured Feedback for Innovation, Balancing Critique and Creativity, Intentions and Practicality in Technical Problem-Solving

**Learning Objective:** Understand and apply the “Two-But Rule” to foster constructive feedback, balance innovation with practicality, and strengthen team collaboration in technical problem-solving.

**Summary:** This chapter explores the “Two-But Rule,” a method for enhancing innovation within technical teams by pairing each critique with a solution-focused alternative. By transforming feedback into a structured, iterative process, the Two-But Rule keeps discussions productive, allowing teams to overcome obstacles without stifling creativity or momentum. Through practical applications, case studies, and best practices, the chapter demonstrates how this approach can help teams achieve more balanced, sustainable innovation.

---

## Introduction

The “Two But Rule,” as introduced by John Walpole, presents a fresh approach to ideation and problem-solving within teams, particularly in technical and innovation-driven environments. By encouraging participants to challenge ideas constructively through the use of “but,” this method transforms negative feedback into a tool for exploration rather than dismissal. Instead of stifling creativity, the two-but framework enables iterative improvement, driving discussions towards more innovative and practical solutions.

In this chapter, we will explore how the “Two But Rule” fosters productive ideation, how it helps overcome common challenges in innovation, and how teams can apply this method to avoid the pitfalls of both overly negative and overly agreeable brainstorming. By understanding the balance between challenge and momentum in discussions, software engineers and innovators can harness this technique to push their ideas further, ultimately leading to more robust and successful projects.

---

## How the Two-But Rule Works

The Two-But Rule is a structured method that encourages constructive feedback and maintains momentum in technical discussions. By requiring each critique to be paired with a potential solution, this approach keeps innovation flowing and helps teams overcome obstacles without stifling creativity.

### Constructive Criticism in Technical Problem Solving

The Two-But Rule is rooted in its ability to create a structured approach to criticism that fosters solution-oriented thinking rather than idea rejection. In technical problem-solving, where teams often face complex and multi-layered challenges, this approach is critical for maintaining momentum while

addressing potential risks or limitations.

Traditional feedback in engineering teams can often center around a single “but” — as in, “But that won’t work,” which quickly halts progress and deflates creative energy. By contrast, the Two-But Rule compels a second step: if you identify a flaw, you must also propose an alternative solution, typically phrased as, “But we could try this instead.” This keeps the focus on refining ideas and prevents conversations from stalling out when obstacles arise.

For example, in a system design discussion, an initial critique might be, “But the current architecture could create a bottleneck at scale.” Instead of leaving the conversation there, the second “but” would offer a path forward: “But we could mitigate that by introducing horizontal scaling earlier in the process.” This dual-step critique enables the team to not only identify potential problems but also to continue moving toward practical solutions.

The Two-But Rule serves as a mechanism to keep discussions dynamic and forward-looking, especially in iterative technical work like software development or system architecture. By combining criticism with solution-finding, the rule encourages active problem-solving while minimizing the risk of shutting down ideas prematurely.

## Using the Rule to Foster Team Innovation

The Two-But Rule plays a crucial role in creating an environment where innovation can thrive, especially in technical teams where creativity must be balanced with practical problem-solving. Innovation often requires pushing through initial hurdles, and the Two-But Rule’s feedback structure facilitates this by creating a space where criticism is welcomed but must always be accompanied by an effort to maintain momentum.

In brainstorming or ideation sessions, the Two-But Rule can be used to encourage rapid iterations of feedback. Walpole, in his podcast discussion, points out that teams often find breakthroughs after around five rounds of “but” iterations. This structured back-and-forth allows ideas to evolve organically, with each round of feedback providing a new avenue for exploration rather than stalling out on perceived flaws.

In a software development team, for instance, one member might propose a new API design that another team member critiques by saying, “But this approach might introduce latency issues.” Using the Two-But Rule, the discussion would not stop there; instead, the feedback would continue with, “But we could reduce latency by caching these calls more efficiently.” This keeps the conversation solution-focused, allowing the team to improve the original idea without dismissing it entirely.

Moreover, structured feedback loops prevent the problem of “paralysis by analysis,” which occurs when teams become bogged down in endless debate without moving forward. The Two-But Rule prevents this by keeping discussions dynamic, encouraging team members to focus on finding workable solutions, even if those solutions are not immediately obvious. The rule ensures that every criticism serves a dual purpose: identifying a challenge and simultaneously pushing the team closer to a resolution.

In practical terms, the Two-But Rule can be integrated into daily operations, such as Agile stand-ups or sprint retrospectives, where team members regularly provide feedback on features, design choices, or processes. By incorporating the Two-But Rule, these meetings can remain constructive and action-oriented, making it easier to arrive at innovative solutions that address real technical concerns.

For more information on how the two-but rule works, plug these search terms into ChatGPT:

- "Two-But Rule in technical problem solving"
- "Constructive criticism vs. single-but thinking in engineering"
- "How to balance critique with solutions in team discussions"
- "Avoiding momentum loss in feedback sessions"

---

## Addressing the Challenges of Feedback and Innovation

Fostering a culture of constructive feedback and innovation in technical teams can be challenging. Teams often struggle to balance necessary criticism with forward momentum. In this section, we explore how the Two-But Rule helps address these challenges by encouraging both critical thinking and solution-focused discussions.

### Overcoming Negative Cultures in Technical Teams

Negative cultures in technical teams often arise when feedback focuses solely on problems without offering solutions. This "single-but" thinking, common in many environments, can stifle innovation and demoralize team members. Common traits of negative cultures include:

- **Premature rejection of ideas:** Engineers may say, "But this will break under heavy load," without offering alternatives.
- **Focus on limitations:** Feedback is centered around what can't be done, rather than how to overcome obstacles.
- **Fear of proposing new ideas:** Team members hesitate to share creative solutions, knowing they will be shot down.

In these environments, criticism halts innovation and ideas are dismissed without further exploration. The Two-But Rule addresses this by requiring:

1. Acknowledging valid concerns – Identify the problem, but don't let the discussion stop there.
2. Offering a potential solution – Follow the concern with an actionable alternative.

For example: Instead of saying, "But this system will crash under high load," use the Two-But approach: "But we could mitigate this with load balancing or automated failover mechanisms."

This method helps technical teams:

- Keep momentum by focusing on solutions.
- Encourage more open dialogue by creating a culture where constructive criticism is valued.

- Prevent ideas from being prematurely discarded.

## Turning Negative Thinking into Positive Solutions

While negative feedback can be a problem, avoiding criticism entirely creates its own set of challenges. In some teams, there's a "no-buts" culture, where feedback is always positive and criticism is avoided to maintain harmony. This can lead to:

- **Poor decision-making** – Bad ideas go unchallenged and are implemented without proper scrutiny.
- **Innovation stagnation** – Teams stop questioning assumptions, leading to a lack of growth or improvement.
- **Regulatory or operational pitfalls** – As seen in the ridesharing and scooter industries, rushing to implement ideas without considering potential downsides can result in logistical chaos and legal issues.

In these environments, the Two-But Rule helps by introducing constructive criticism in a structured way. To ensure feedback remains productive:

1. Start with a positive note or acknowledgment in order to prevent the team from feeling discouraged.
2. Identify the problem clearly, ensuring that the concern is valid and relevant.
3. Propose a solution, shift the conversation from a problem-focused discussion to a solution-driven one.

For example, rather than saying, "But this project is too expensive," you could say, "But we could reduce costs by outsourcing certain tasks or using open-source tools."

Key Techniques to Foster Positive Solutions:

- **Encourage brainstorming with structured feedback loops** – After each idea, identify a concern and immediately follow it with a potential fix.
- **Limit "analysis paralysis"** – Keep feedback sessions short and goal-oriented to avoid over-analyzing the problem without moving toward action.

Incorporating the Two-But Rule ensures that even when challenges are identified, the team remains focused on how to move forward, fostering a dynamic and innovative environment.

---

## The Role of Intentions in Innovation

A critical component of fostering innovation within technical teams is understanding the intentions behind ideas. It's not just about what ideas are presented but why they are suggested in the first place. Understanding the motivation behind suggestions helps create a more effective and collaborative environment, allowing teams to align their goals and create better solutions.

## Understanding Intentions Behind Ideas

In technical discussions, there is often a gap between what someone proposes and the underlying intention driving that suggestion. Recognizing these intentions is key to fostering innovation because it helps teams align around common goals and address root causes rather than surface-level problems. Here's how understanding intentions can lead to better outcomes:

**Clarify motivations:** Ideas are often rejected prematurely because the underlying motivation isn't clear. By understanding the "why" behind a suggestion, teams can better evaluate its potential.

- Example: A developer might suggest rewriting a legacy system. While this sounds costly, the intention might be to improve system security or scalability.

**Enable better collaboration:** When teams understand the driving intentions behind ideas, they are more likely to work together toward a common goal. This can foster a stronger sense of ownership and collaboration.

- Example: If a team member's intention is to reduce technical debt, others can rally around this goal by proposing incremental updates rather than rejecting a full system overhaul.

**Uncover hidden opportunities:** Often, the intention behind a suggestion reveals opportunities that weren't immediately apparent.

- Example: Proposing a new tool may seem unnecessary at first, but the intention may be to improve developer productivity, reducing long-term costs.

Understanding the intentions behind ideas can help teams avoid dismissing valuable input too quickly and encourages deeper exploration of the problems being solved.

## How to Uncover Hidden Intentions in Technical Discussions

Uncovering the true intentions behind ideas requires active listening and asking the right questions. Here are some practical techniques to help technical teams reveal and understand intentions more effectively:

**Ask open-ended questions:** Encourage team members to explain the reasons behind their suggestions.

- Example: Instead of simply asking, "Why are you suggesting this?" try "What problem are you hoping to solve with this suggestion?" or "How do you see this improving our current process?"

**Practice empathy and active listening:** Create a culture where team members feel heard and understood by focusing on what is being said rather than immediately forming a response.

- Listen carefully to the feedback or objections, and then summarize the underlying intention: "It seems like you're concerned about scalability. Is that your main motivation here?"

**Identify competing intentions:** It's possible that multiple ideas or feedback from different team members represent conflicting goals. Highlight these and work toward a unified solution.

- **Example:** One team member might want to optimize for performance, while another focuses on minimizing costs. Understanding both intentions allows the team to brainstorm solutions that address both concerns.

Key Tools to Facilitate Understanding Intentions:

- **Clarification Rounds:** After an idea is proposed, allow each team member to ask questions aimed at clarifying the motivation behind the suggestion.
- **Empathy Mapping:** Use tools like empathy maps to visualize the needs and goals of different stakeholders in the discussion, helping teams keep the broader context in mind.
- **Role Reversal Exercises:** Have team members argue from the perspective of others to better understand different intentions and align on shared goals.

By uncovering the intentions behind feedback and suggestions, teams can engage in more meaningful discussions, leading to more innovative and well-rounded solutions. This approach helps technical teams align their goals, ensuring that innovation is driven not just by ideas but by a shared understanding of the “why” behind them.

---

## Balancing Innovation and Practicality in Technical Teams

In technical teams, it's essential to strike a balance between fostering innovative ideas and ensuring that they are grounded in practical, scalable solutions. While innovation drives growth, too much focus on disruptive change can lead to unintended consequences. In this section, we'll explore how to balance creativity with practicality and avoid the pitfalls of overcomplicating technical solutions.

### Avoiding Bad Innovation: The Problem with Disruptive Innovation

Disruptive innovation often leads to breakthroughs, but it can also cause harm if not carefully regulated. Here are some key risks associated with unregulated innovation:

Unregulated Innovation Risks:

- **Regulatory Arbitrage:** Disruptive technologies that ignore regulations often face backlash. For instance, ridesharing apps initially bypassed local transportation rules, only to later face stricter regulations after disrupting the taxi industry.
- **Negative Consequences:** Overlooking practical concerns can lead to long-term damage. The scooter-sharing industry, for example, experienced rapid growth followed by city-imposed restrictions due to public safety concerns.

Radical Improvement vs. Disruption:

- **Radical Improvement:** Instead of focusing on purely disruptive innovation, teams should aim for radical improvement—gradual, thoughtful enhancements to existing systems that avoid excessive risks.

- **Balanced Approach:** Radical improvement allows teams to innovate without destabilizing the industry or running into major legal challenges.

Examples:

- **Ridesharing:** Early rideshare companies embraced disruptive innovation but encountered problems when they disregarded regulatory requirements.
- **Blockchain:** While blockchain introduced new solutions, it also faced scalability and legal issues due to a lack of foresight on how to integrate with existing systems.

Practical Ways to Avoid Bad Innovation:

- **Conduct Feasibility Studies:** Before diving into a new idea, evaluate the potential risks, costs, and technical challenges.
- **Be Mindful of Regulations:** Collaborate with legal and regulatory experts to ensure compliance while innovating.
- **Prioritize Usability and Scalability:** Keep the end user in mind, ensuring that any new solution is easy to implement, maintain, and scale.

## Moving from Ideation to Practical Implementation

Once a team has a great idea, the next challenge is turning that idea into a working solution without falling into the trap of overcomplicating things. Here's how to transition from concept to execution effectively:

Steps to Implement Innovation:

1. Divide the idea into manageable pieces and prioritize which components to implement first.
2. Create a simple prototype to test the core concept before investing time and resources into full development.
3. Iterate and Improve: Gather feedback on the prototype, make necessary adjustments, and gradually refine the solution.

Managing Technical Debt:

- **Regular Code Reviews:** Ensure that new features don't pile up technical debt by maintaining clean, scalable code.
- **Continuous Refactoring:** Encourage a culture of ongoing refactoring to prevent future code maintenance problems.

Avoiding Overcomplication:

- **KISS Principle:** Keep it simple. When designing new features or systems, avoid adding unnecessary complexity that could bog down future work.
- **Evaluate Needs:** Before adding new services or tools, ensure they truly solve a problem without creating new operational headaches.

## Using the Two-But Rule to Keep Innovation Grounded

The two-but rule is essential for balancing innovation with practicality. By addressing challenges and proposing alternatives, teams can push ideas forward without stifling creativity.

1. **First But:** Identify a challenge with the idea ("But that won't work because...").
2. **Second But:** Propose a solution or alternative ("But we could adjust it by doing...").

This back-and-forth method helps teams iterate quickly and refine ideas in a constructive manner. For example:

- Scenario: A team proposes a new API that is difficult to integrate.
- First But: "But it will take too long to integrate with our current systems."
- Second But: "But we could simplify it by focusing on core functionalities first."

For more information on Innovation and practicality, plug these search terms into ChatGPT:

- "What is regulatory arbitrage in tech?"
- "Radical improvement vs. disruption in software"
- "Prototyping and iteration in software development"
- "Managing technical debt in iterative projects"
- "KISS principle in technical design"

---

## Applying the Two-But Rule in Software Engineering

The two-but rule can be particularly useful in software engineering, where technical challenges and complex systems often require collaborative problem-solving. This section explores how to implement the rule in daily workflows, including Agile processes and team feedback cycles, and provides a real-world case study to demonstrate its effectiveness.

### Implementing Structured Feedback in Agile Teams

Agile and DevOps teams thrive on feedback loops to improve processes and solutions iteratively. The two-but rule fits naturally into these environments, helping teams address problems and generate solutions without stalling progress.

How to Apply the Two-But Rule in Agile Settings:

- **Daily Stand-ups:** Use the two-but rule during stand-up meetings to identify blockers and propose solutions.
  - Example: "We're behind on the API integration \*but\* we could adjust our timeline \*but\* that might impact the next sprint."

Sprint Planning:

- During **sprint planning**, apply the two-but rule to assess the feasibility of tasks. For example, if a feature seems too complex, propose alternatives.
  - Example: "This new feature requires a third-party integration *\*but\** we could reduce complexity by building a simpler version first."

#### Retrospectives:

- **Retrospectives** are ideal for reflecting on challenges. Encourage team members to express concerns constructively.
  - Example: "The testing pipeline failed frequently last sprint *\*but\** we could resolve this by automating test case generation *\*but\** it may take more time."

#### Code Reviews:

- **Code reviews** are another opportunity to apply the two-but rule. Instead of simply rejecting code, reviewers can suggest improvements.
  - Example: "This method is too slow for high-volume traffic *\*but\** we could optimize it by caching results *\*but\** we'll need to add a timeout mechanism."

#### Fostering a Culture of Constructive Feedback:

- Ensure team members feel comfortable presenting concerns and suggestions.
- Frame feedback in a way that acknowledges problems but focuses on actionable solutions.
- Leverage the iterative nature of Agile to revisit and refine solutions with each sprint.

## Case Study: Applying the Two-But Rule in a Software Development Team

Let's look at a real-world example of how a software development team used the two-but rule to resolve a complex technical issue.

Scenario: A team developing a web application faced challenges with integrating a new payment gateway. The gateway's API was more complex than expected, and the integration was causing delays in the project timeline.

- **Initial Discussion:**
  - Team Lead: "We're behind schedule on the payment gateway integration *\*but\** we need to find a way to reduce the complexity *\*but\** without compromising security."
- **First Round of Feedback:**
  - Developer 1: "But the API requires extensive token handling, which is slowing down our process *\*but\** we could simplify it by using an existing library for token management."
  - Developer 2: "But that library has limited support for our authentication method *\*but\** we could create a wrapper around the library to make it compatible."
- **Second Round of Feedback:**
  - Team Lead: "But that wrapper might introduce new bugs *\*but\** we could create a testing suite to handle edge cases during the integration."

- **Outcome:**
  - After several iterations of the two-but rule, the team decided to implement the wrapper with thorough testing, allowing them to complete the integration without further delays or security risks.

For more information on how to apply the two-but rule in software engineering, plug these search terms into ChatGPT:

- "Applying Two-But Rule in sprint retrospectives"
- "Two-But Rule in Agile stand-ups and sprint planning"
- "Constructive feedback loops in DevOps and Agile"

---

## Best Practices for Fostering Innovation in Technical Teams

Cultivating a culture of innovation within technical teams requires deliberate strategies that balance openness with constructive feedback. This section outlines best practices for encouraging creativity while ensuring ideas are critically assessed in a supportive environment.

### Creating a Culture of Openness and Collaboration

For innovation to thrive, team members must feel comfortable sharing their ideas without fear of judgment or rejection. This involves fostering psychological safety and building a culture where every idea can be heard and improved upon.

#### Encourage Open Dialogue:

- Create regular opportunities for team members to share ideas in open forums, such as brainstorming sessions or retrospectives.
- Example: In Agile teams, daily stand-ups or sprint reviews are ideal moments for surfacing new ideas.

#### Normalize Constructive Criticism:

- Encourage a feedback culture where constructive criticism is seen as a way to improve ideas, not shut them down.
- Introduce the **two-but rule** to balance feedback by pairing criticism with actionable alternatives.

#### Promote Psychological Safety:

- Leaders should model vulnerability and openness, demonstrating that it's okay to share unfinished or imperfect ideas.
- Make it clear that failure is a part of the innovation process.

#### Facilitate Inclusive Participation:

- Ensure that all team members, regardless of seniority or role, feel empowered to contribute.

- Techniques such as anonymous idea submissions can encourage participation from more reserved team members.

## Avoiding Stifled Innovation: Encouraging Playfulness in Problem Solving

Allowing room for experimentation and playfulness can stimulate creative breakthroughs. Teams need to feel they have the freedom to explore unconventional approaches without being penalized for failure.

- **Embrace “Safe Failure”:**
  - Create environments where it’s safe to fail early and often. This is particularly important in prototyping phases or R&D initiatives.
  - Example: Implement fail-fast strategies in software development to allow quick pivots when ideas don’t pan out.
- **Iterative Problem-Solving:**
  - Break complex problems into smaller, more manageable tasks that can be iteratively explored.
  - Example: Use Agile sprints or Kanban boards to ensure that ideas are tested and refined over short, measurable periods.
- **Encourage Cross-Disciplinary Play:**
  - Facilitate sessions where team members from different backgrounds collaborate, encouraging diverse perspectives.
  - Example: Pair front-end developers with DevOps engineers to explore different ways of solving deployment issues.

---

## Conclusion

The two-but rule provides a structured and effective approach for fostering innovation in technical teams, especially within software engineering environments. By encouraging constructive feedback—where a problem is always paired with a potential solution—the rule ensures that momentum is maintained during ideation and problem-solving sessions. This structure helps to avoid both the stifling effects of overly negative feedback and the stagnation of “no-buts” cultures, where necessary criticism is avoided.

Throughout this chapter, we have explored how the two-but rule can be implemented to enhance collaboration, address technical challenges, and ultimately drive innovation. Whether in Agile stand-ups, sprint planning, or during retrospectives, the two-but rule creates a balance between critique and creativity, allowing teams to navigate complex technical problems without losing momentum.

By applying this framework, software engineers can turn negative thinking into positive solutions, leading to more robust technical designs, better team dynamics, and more sustainable innovation. As teams continue to adopt and adapt the two-but rule in their workflows, they will find that it not only

improves their technical output but also fosters a more open, playful, and innovative work environment.

---

## Discussion Prompts

### 1. **Balancing Criticism and Creativity:**

1. How can teams find the right balance between constructive criticism and open ideation? What are some potential pitfalls when this balance shifts too far in either direction?

### 2. **Overcoming Negative or “No-Buts” Cultures:**

1. Have you experienced a “no-buts” or overly critical culture in a team? How did it affect innovation, and what steps could be taken to shift toward a Two-But approach?

### 3. **Understanding Intentions Behind Ideas:**

1. How can uncovering the intentions behind suggestions improve team collaboration and idea development? What techniques have you used or observed that were effective in understanding the “why” behind ideas?

### 4. **From Ideation to Practical Implementation:**

1. When translating innovative ideas into real-world solutions, what are some ways teams can avoid overcomplicating their designs? How does the Two-But Rule help in identifying practical next steps?

### 5. **Creating a Culture of Open Feedback:**

1. How can leaders foster an environment where team members feel safe sharing their critiques and ideas? What role does psychological safety play in innovation, and how can it be strengthened in technical teams?

### 6. **Reflecting on the Two-But Rule’s Long-Term Impact:**

1. How might the Two-But Rule impact team dynamics and innovation over time? What are some potential long-term benefits or challenges when consistently applying this approach?