The Feedback Language Swipe File

For Engineering Managers: What to Say, When to Say It, and How

leadingwithquality.com | Stuart Ashman

Most managers don't avoid feedback because they don't care.
They avoid it because they don't know what to say.

This swipe file gives you the exact language - and the structure to use it in.
Keep it close the first few times. After that, it becomes instinct.

---

1 - The Core Language Shift: Labels vs. Observations

The single most common feedback mistake is using labels instead of observations.
Labels ("you never," "your code is always") describe a person's character.
The other person immediately looks for the exception - and once they find it, they've stopped listening.

Observations describe what you actually saw. They're specific, dateable, and something the other person can respond to - or add context to.

The shift:

Instead of this (label)	Say this (observation)
"You never test your code properly."	"I noticed the last three PRs broke the build on merge."
"You always go off-spec."	"I observed that the feature shipped with two additional components we didn't scope."
"Your estimates are always wrong."	"I noticed your last four estimates came in at roughly 2x the actual time."
"You're not communicating well with the team."	"I noticed the team was unaware of the API change until the day before the sprint review."

"Your code reviews are too slow."	"I observed that three PRs from last sprint were open for more than five days before you reviewed them."
"You don't seem engaged."	"I've noticed you haven't contributed in the last three architecture discussions."

Your go-to phrases:

- "I noticed..."
- "I observed..."
- "I have seen..."
- "What I observed was..."

When in doubt, describe the specific output, behaviour, or event - not the person behind it.

---

2 - Two Types of Feedback: Reinforcing and Change-Focused

Feedback is not praise or criticism. It is information.

Reinforcing feedback tells someone what to keep doing - specifically enough that they can repeat it.

Change-focused feedback tells someone what to adjust - specifically enough that they know what different looks like.

Neither is a compliment. Neither is a rebuke. Both are data.

Reinforcing Feedback - Engineering Examples

❌ "Great work on that feature."
✅ "The way you structured the rollout plan - phased by risk level, with rollback documented at each stage - made it easy for the on-call team to follow. Do that again."

❌ "Good job in the incident review."
✅ "In the incident review, you named the contributing factors before the root cause - that framing kept the conversation diagnostic rather than defensive. That was the right call."

❌ "Your code quality is really improving."
✅ "I noticed your last five PRs had zero review cycles past the first round. The logic was clear, the tests covered the edge cases, and the commit messages explained the why. That's the standard."

❌ "Nice work with the new engineer."
✅ "I saw how you paired with Jamie on the auth refactor - you explained the reasoning behind the pattern choice, not just the pattern. That's how you build judgment in a team, not just skill."

Change-Focused Feedback - Engineering Examples

❌ "Your estimates are unreliable."
✅ "Over the last two sprints, you committed to 13 points and delivered 5. I want to understand what's happening - are the estimates off, or is something getting in the way mid-sprint?"

❌ "You need to communicate more."
✅ "I noticed the backend team was blocked for two days waiting on your API schema. They didn't know you were still working on it. I need you to flag that kind of dependency earlier."

❌ "Your pull requests are hard to review."
✅ "I noticed your last PR had 47 files changed with one commit message. That makes it very difficult for reviewers to understand the intent. Going forward, I need smaller, scoped commits with clear messages."

❌ "You need to be more collaborative."
✅ "I observed that in the last three design discussions, you presented a solution and didn't ask for input. A couple of people told me afterward they had concerns they didn't raise. I need you to create more space for pushback in those conversations."

---

3 - The 4-Step Feedback Conversation

Use this structure for any feedback conversation - reinforcing or change-focused.

Step 1: Create an opening
Ask for a few minutes. Don't drop feedback mid-standup or in a Slack message.
"Got ten minutes? I want to share something with you."
If they're busy: "When's a good time? It's not urgent, but I don't want to leave it too long."

Step 2: Describe the behaviour or result
Use your observation language from Section 1.
Stick to what you actually saw - not what someone told you, not your interpretation of intent.
One specific example is more powerful than a general pattern.

Step 3: State the impact - using "I" language
This is not about blaming. It's about showing why the observation matters.
"When that happens, I can't give the team an accurate picture of where we are."
"That means the on-call engineer is going in blind at 2 am."
"It tells me something is getting in the way that I don't know about yet."

Step 4: Make a request
Be specific about what you need to happen differently - or what you want to continue.
"What I need from you going forward is..."
"What I'd like you to keep doing is..."
"Can you walk me through what happened? I want to understand before we work out what changes."

Note on Step 4: If you don't know what you want, don't skip to the request.
Ask what they need from you. You may not have the full picture yet.

---

4 - Timing Rules

Feedback gets its power from proximity.
The closer it is to the observed moment, the more useful it is.

The order of preference:

1. Same day or next day - as close to the event as possible
2. Within the same sprint or work cycle
3. Your next scheduled 1:1
4. ⚠️ After a delay - always say so: "I should have raised this sooner."
5. ❌ Annual review - far too late to be actionable; only ever a summary

What to do when a 1:1 runs out of time:

Don't bump the feedback to next week. Reschedule a separate 15-minute slot this week.
Feedback is not an agenda item that can wait. If it can wait indefinitely, ask yourself whether you actually intend to give it.

The "why didn't you tell me sooner?" test:
If the person you're giving feedback to would reasonably ask this, you've waited too long.

---

5 - Public vs. Private

The rule you were probably taught: Praise in public. Criticise in private.

The actual rule: All feedback in private first - including praise.

You don't know how someone will receive feedback until you give it.
Some people find public recognition energising. Others find it mortifying.
Assuming you know which camp someone is in without asking is the mistake.

What to do instead:

Before a team meeting or all-hands, ask:
"I want to recognise what you did on this in front of the team. Are you comfortable with that?"

Team-level praise vs. individual praise:
Most people are comfortable with team-level recognition ("this team shipped something hard"). Individual singling-out in front of peers is where preferences diverge most.

Establish this in your first 1:1 - see the Feedback Contract template for the exact questions.

Quick Reference Card

Core language: "I noticed..." / "I observed..." / "I have seen..."

Two types: Reinforcing (keep doing this) | Change-focused (adjust this)

4 steps: Opening → Observation → Impact → Request

Timing: Same day > this week > next 1:1 > never treat as bumpable

Public vs. private: Ask first. Always private first, including praise.

Stuart Ashman
Engineering Leadership Coach | Leading with Quality
leadingwithquality.com

Download the companion Feedback Contract template and more resources at leadingwithquality.com/ → resources