

How to build a feedback loop that improves itself.

UNIFY · UNDERSTAND · ACT · IMPROVE

Four stages, any stack. Copy the structure, plug in the tools you already use, and run the loop once in 90 days.

INSIGHTS FROM

Canva

 Notion

ElevenLabs

 Wispr

Contents

START HERE

A pipeline ends. A loop improves.	03
--	----

THE RUBRIC

The Loop Maturity Model	04
-------------------------------	----

Add it up: scoring your loop	05
------------------------------------	----

THE FOUR STAGES

01 · Unify	06
------------------	----

02 · Understand	08
-----------------------	----

03 · Act	10
----------------	----

04 · Improve	12
--------------------	----

Copy the structure: the loop on one page	14
--	----

WHAT IT COMPOUNDS INTO

Two companies that look identical	15
---	----

Where loops break	16
-------------------------	----

Who owns the loop	17
-------------------------	----

Walk one problem through all four stages	18
--	----

Thirty, sixty, ninety	19
-----------------------------	----

A pipeline ends. A loop improves.

Most companies are not short on customer feedback. It arrives every hour, from a dozen surfaces, in volumes no team can read. The problem is what happens next: most of it never becomes a decision, a fix, or a change the customer can feel.

Most programs are built as pipelines. Feedback goes in, a report comes out, and the work stops there.

A loop keeps going. It checks whether the fix moved the signal, tells the customers who asked, and carries what it learned into the next cycle.

The stakes just rose. Agents and assistants now answer for the business at the speed of software. Feed them scattered, contradictory feedback and they will be confidently wrong in front of customers.

A loop improves itself when every cycle leaves something behind: a sharper taxonomy, a clearer owner, a cleaner signal for the next one.

01

Unify

Pull every source into one place.

02

Understand

Group it by theme, then size it by who is asking.

03

Act

Give every output one owner.

04

Improve

Check the fix moved the signal. Then tell the customers who asked.

The Loop Maturity Model

Twelve questions about the loop as it runs this week, not the one on the roadmap. Score it on the next page.

	NO 0	PARTLY 1	YES 2
UNIFY ----- DATA			
01 Every channel lands in one place without anyone copying it there.	☐	☐	☐
02 Feedback is tied to the account, plan and revenue behind it.	☐	☐	☐
03 Someone owns the taxonomy, and it matches today's product.	☐	☐	☐
UNDERSTAND ----- EVIDENCE			
04 Themes are named in your customers' words, not a generic category list.	☐	☐	☐
05 Issues are sized by distinct customers and revenue, not mentions.	☐	☐	☐
06 You can tell a representative theme from a loud minority.	☐	☐	☐
ACT ----- ACTION			
07 A theme crossing a threshold triggers work without anyone running a report.	☐	☐	☐
08 Every output, from roadmap item to risk alert, has one named owner.	☐	☐	☐
09 Insight reaches whoever can resolve it in hours, not weeks.	☐	☐	☐
IMPROVE ----- MEMORY			
10 You check whether a fix moved the signal, not just whether it shipped.	☐	☐	☐
11 Customers who reported an issue hear when the fix ships.	☐	☐	☐
12 Prior findings and ownership carry into the next cycle automatically.	☐	☐	☐

Add it up

YOUR TOTAL _____ / 24

Two points for yes, one for partly, none for no.
Count honestly: a question you would need a week to answer is a no.

0-7

You are collecting feedback, not using it.

Five teams hold five partial versions of the truth, and decisions run on whoever spoke last.

Next: get every channel into one deduplicated stream tied to accounts.

8-14

You can see the feedback but cannot move on it.

You know what customers said. You cannot say how many, how much revenue, or who owns the fix.

Next: weight themes by distinct customers and revenue, and give every output one owner.

15-20

The loop runs because people push it.

Every step works, and every step needs a person to make it happen. Throughput is capped by their week.

Next: automate the trigger and the signal check, the two steps that quietly consume the most human time.

21-24

The loop improves itself.

Findings, ownership and context carry forward, so each cycle costs less than the last and the AI front line stays grounded.

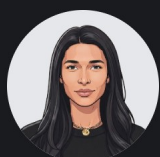
Next: widen it. More channels in, more teams served from the same truth.

Most teams score between 8 and 14: the data is in one place, the sizing is rough, and nothing routes itself. Whatever your score, your AI front line cannot outrun it.

Unify

Pull every source into one place: tickets, calls, reviews and community, deduplicated, tied to the accounts and revenue behind them, under a taxonomy that reflects how your customers actually talk.

“Beyond just keywords, we can actually understand what our users are saying across every interaction we plug in.”



Emma Auscher

Global VP of Customer Experience, Notion



Notion

01 UNIFY · THE PRACTICES

Five teams hold five conflicting versions of the truth. Every later stage, and every AI system, inherits the quality of this foundation.

01 / Centralize every channel

Canva unifies 13,841 feedback themes across 34 sources, from Zendesk to Gong to Reddit to Discord, so a marketer, a PM and a support lead query the same reality.

02 / Normalize the data

A dozen spellings of “USA”, timestamps as raw epochs, the same person a User ID here and an Author there. Standardising those fields is what makes segmenting possible at all.

03 / Tie feedback to revenue

BuildOps ties feedback to 128,000+ accounts carrying ARR, segment, churn risk and assigned CSM. A complaint arrives with a dollar figure and an owner attached.

04 / Build the taxonomy

Philo classifies 319,000+ insights into a hierarchy you navigate rather than excavate. This is the shared language every team and every model reads from.

This is maintenance, not setup. At Western Union, someone renamed a set of taxonomy keywords and every downstream report and automation built on the old mappings silently broke. No error message. The data just quietly began to mislead. A loop that improves itself keeps its taxonomy current as the product changes.

THE TEST

Pick one complaint from this morning. In under a minute, can you see everywhere else that customer has raised an issue, tied to their account and revenue?

Understand

Group it, then size it by who is asking: how many customers are affected, which segments, how much revenue sits behind it, and whether a theme is widespread or merely loud.

“We surface product insights in days, not weeks. Interpret aligns everyone around the top issues and requests.”



Jesse Walker

Head of Insights and User Advocacy, Canva

Canva

02 UNDERSTAND · THE PRACTICES

The default way companies prioritise is that the loudest voice wins. Measurement replaces that, fast enough that the answer arrives before opinion hardens into a decision.

01 / Group by theme, in their words

Name themes the way customers describe them. “Checkout” breaks into Apple Pay fails, promo code and slow load, and each one can be sized and owned on its own.

03 / Catch what instrumentation misses

Logs tell you what your software did, not which bug never threw an error. One AI coding tool surfaces memory leaks that trigger crashes; a consumer app finds crashes during photo upload.

02 / Size by impact, not mentions

Ten furious power users repeating a complaint forty times is not a theme shared by ten thousand customers. Weight by distinct customers and ARR, and the order changes.

04 / Read commercial risk early

Feedback carries churn and pricing signals before revenue does: a map of where users hit the paywall and resent it.

RANKED BY MENTIONS

Slow load



Promo code



Apple Pay fails



RANKED BY CUSTOMERS × ARR

Apple Pay fails



Slow load



Promo code



Illustrative. The loudest theme is rarely the costliest.

THE TEST

Can you state, as a number you would defend in a meeting, how many customers an issue affects and how much revenue is exposed?

Act

Give every output one owner. When a theme crosses a threshold, the work lands with the people and systems that resolve it, your AI front line is grounded on it, and nothing waits for a quarterly review.

“It elevates what our support engineers are capable of. They identify root causes and hand engineering issues that are already actionable.”



Samm Yates

Founding Head of CX, Wispr Flow



03 ACT · THE PRACTICES

Understanding without action isn't neutral, it's corrosive. Every time a customer speaks and nothing changes, they learn that speaking up is pointless.

01 / Set the trigger in advance

Decide what crosses the line: a theme doubling in a week, a risk signal on a top account. PMs at one creator platform get alerted to a spike in their own area within hours, and nobody has to remember to run the report.

03 / Route it to where the work happens

One communications app links themes straight to engineering tickets, so a one-way-audio bug arrives with the complaints attached and the squad identified.

02 / One output, one owner

Roadmap items go to product, risk alerts to success, bug fixes to engineering, the weekly digest to leadership. Shared ownership is how work goes unowned.

04 / Ground and check your AI front line

QA used to mean hand-reading two percent of tickets and hoping it generalised. When a machine answers, a sample that size is negligence.

Roadmap

OWNER · PRODUCT

JIRA

LINEAR

Risk alerts

OWNER · SUCCESS

SALESFORCE

SLACK

Bug fixes

OWNER · ENG

JIRA

PAGERDUTY

Weekly digest

OWNER · LEADERSHIP

SLACK

EMAIL

THE TEST

Take the last theme that crossed your threshold. Did work start without anyone asking, and can you name the one person who owned what happened next?

Improve

Did the fix move the signal? A loop with a memory checks, tells the customers who asked, and leaves the next cycle easier and more accurate instead of starting from zero.

“If you’re not operating from the same customer understanding, within a month product, marketing and engineering are talking about completely separate things.”



Tanmay Jain

Product & Growth Lead, ElevenLabs

ElevenLabs

Most feedback work stops when the fix ships. Nobody checks whether the signal moved, nobody tells the customer, and the next cycle starts from zero.

01 / Monitor the signal, not the ticket

When a fix ships, Interpret watches the theme drop: resolved from the customer's side, not merely marked "done."

"Having to give it all of our ownerships and topics every single time to explain who I am as a team is a little frustrating."

PM · EUROPEAN FINTECH

02 / Tell the customers who asked

The same check names the accounts that reported it, so the follow-up goes to people who will notice.

"I don't think the automation has the ability to self update itself to learn what it already checked."

POWER USER · CONSUMER HARDWARE

A loop that does not remember does not improve. It only repeats.

03 / Keep the artifact

A new theme stays in the taxonomy. A resolution is recorded against the accounts involved.

04 / Feed it forward

Ownership, prior findings and known failure modes carry from one run to the next.

05 / Let the front line learn

Every wrong AI answer caught becomes context that prevents the next one.

Copy the structure. Plug in the tools you already use.

Four stages and one trigger. The tools are interchangeable; the order is not.

01 · UNIFY

Pull every source into one place.

Tickets	ZENDESK	INTERCOM	Calls	GONG	ZOOM
Reviews	APP STORE	NPS	Community	REDDIT	SLACK

02 · UNDERSTAND

Group it by theme, then size it by who is asking.

Themes named in your customers' words. Sized by customers × ARR, not mentions.

● TRIGGER: THEME CROSSES A THRESHOLD

03 · ACT

Give every output one owner.

Roadmap → Product	JIRA	LINEAR
Risk alerts → Success	SALESFORCE	SLACK
Bug fixes → Eng	JIRA	PAGERDUTY
Weekly digest → Leadership	SLACK	EMAIL

04 · IMPROVE

Did the fix move the signal? Then tell the customers who asked.

DETECT → TRIAGE → DO → MONITOR → BACK TO UNIFY

Start with **one loop, one owner, one metric**. Then add the next.

Two companies that look identical

Both have dashboards, a VoC team, a weekly report, AI in support, and an executive who sincerely calls the company customer-obsessed.

FEEDBACK AS A PIPELINE

Collected, sometimes analysed, rarely acted on, never checked afterward. Its AI answers from that same untrustworthy context with no one checking, so customers go quiet and the AI misfires at scale. Every cycle starts from zero.

FEEDBACK AS A LOOP THAT IMPROVES ITSELF

One maintained taxonomy, sized by customers and revenue, triggered at a threshold, routed to one owner, checked against the signal, reported back to the customers who raised it. Its context gets richer every cycle.

In any given week the gap looks small. Then a competitor running the loop is ahead on an axis you never measured, by a distance you cannot close in a quarter.

Where loops break

None of these announce themselves. Each shows up first as something small enough to explain away.

01

The taxonomy drifts

Someone renames a set of keywords and every report built on the old mapping quietly keeps running.

YOU'D NOTICE IT AS

Two teams quote different numbers for the same theme, and both can defend theirs.

02

Mentions stand in for customers

Volume is counted as demand, so ten power users repeating a complaint outrank a thousand quiet ones.

YOU'D NOTICE IT AS

A fix ships against the top-ranked request and the underlying volume barely moves.

03

Insight stops at the report

No trigger and no owner, so the analysis is circulated but never lands in a ticket, a queue or a system.

YOU'D NOTICE IT AS

A weekly readout everyone reads, and no piece of work that cites it.

04

Nobody checks the signal

The fix ships, the ticket closes, and no one looks at whether the theme dropped or tells the accounts that reported it.

YOU'D NOTICE IT AS

You can name last month's feedback-driven fix, but not whether complaints fell, or one customer who heard about it.

05

Every cycle starts from zero

Ownership and prior findings are re-established by hand each time.

YOU'D NOTICE IT AS

Someone, or something, re-explains your product area from scratch every week.

Who owns the loop

Feedback that feeds the roadmap and the AI front line is too consequential to sit between teams. No new title required: the responsibilities follow the four stages.

NAME THE OWNER FOR EACH

STAGE	RESPONSIBILITY	OWNED	SHARED	NO ONE
UNIFY	Own the pipes and the taxonomy. One normalized stream, tied to accounts and revenue, under a hierarchy that matches the product.	☐	☐	☐
UNDERSTAND	Size demand, cost and risk. Defensible numbers on what customers ask for and where revenue is exposed.	☐	☐	☐
UNDERSTAND	Separate representative from vocal. Keep the loud minority from shipping as the actual market.	☐	☐	☐
ACT	Set the triggers. Decide what crosses the threshold, and where the work lands when it does.	☐	☐	☐
ACT	Own each output, and ground the AI front line. Roadmap, risk alerts, bug fixes, digest: one name each. Audit AI answers at full coverage.	☐	☐	☐
IMPROVE	Monitor the signal and close it back. Confirm the fix from the customer's side and tell the accounts that reported it.	☐	☐	☐
IMPROVE	Carry each cycle forward. Ownership and prior findings feed the next run; the front line learns from every miss.	☐	☐	☐

If more than two rows land in **Shared** or **No one**, the loop doesn't have an owner. It has volunteers, and it will stall at the stage nobody is accountable for.

Walk one problem through all four stages

Pick one real problem your customers reported in the last ninety days.

Wherever the walk stalls is your constraint.

01 UNIFY

Can you see every customer who raised this issue, across every channel, tied to their account and revenue, in one place?

02 UNDERSTAND

Can you state how many customers and how much revenue it affects, and whether it's representative or merely loud?

03 ACT

Did crossing the threshold start the work on its own? Did one named owner pick it up, with your AI front line grounded on it?

04 IMPROVE

Did the signal move after the fix? Were the customers who reported it told? Did the cycle leave your taxonomy and context sharper for the next problem?

For most teams the walk stalls at Improve: the step most teams skip, and the one that makes every later cycle cheaper.

Thirty, sixty, ninety

Every stage depends on the one before it, so the order is not negotiable. What is negotiable is scope: one product area, done properly, beats the whole company done thinly.

FIRST 30 DAYS

Unify one product area

Connect the two or three sources that carry most of your feedback, deduplicated into one stream.

Stand up the taxonomy on your own language, not a generic category list.

Tie records to accounts, ARR and an owner.

DAYS 30-60

Size it, then route it

Weight every theme by distinct customers and revenue, never by mentions.

Set one threshold that starts work on its own.

Give each output one owner, in the tool they already use.

DAYS 60-90

Check it and compound it

After each fix, confirm the signal moved, from the customer's side.

Tell the accounts that reported it.

Carry ownership and prior findings into the next cycle.

EXIT TEST

Take one complaint from this morning and see everywhere else that customer has raised it, in under a minute.

EXIT TEST

Bring a number you would defend in a prioritisation meeting, and have it change the order of the work.

EXIT TEST

Name a fix, the theme that dropped after it shipped, and the customers who were told.

Start with **one loop, one owner, one metric**. Then add the next. Ninety days is enough to run it once, end to end, and that is the only proof that matters.



This is the loop we build.

Interpret unifies every channel of customer feedback into one adaptive taxonomy built on your own language, tied to your accounts and revenue. Then it puts that context where your team already works, and where your agents can read it.

TRUSTED BY TEAMS AT

ElevenLabs

 **descript**

 **Fanatics**

 **Wispr**

Canva

 **Notion**

STRAVA

 **Apollo**

UNIFY

Every source, deduplicated and tied to accounts, in weeks not quarters.

UNDERSTAND

Your categories, quantified by customers and revenue, not a model guessing.

ACT & IMPROVE

Automations that fire on a threshold, route to one owner, and confirm the signal moved.

See it on your own feedback.

We connect a couple of your sources and stand up your taxonomy on your real data. By the end you'll know whether this is real: on your feedback, not a demo.

BOOK A DEMO

INTERPRET.COM

