Practice brief

Customer quick response councils

lateralworks FTTM practice series | Voice of the customer | Revision 2, September 2026

Put three to five working users of the product inside each project team and keep them there through the first field season. Most companies already have something like this forming informally around one or two projects. The mechanism to make it run across a portfolio is a weekly one-hour standing meeting per council, a Slack channel for rapid asynchronous response, and council touchpoints written into the project schedule. lateralworks has used this variation of the customer council on many programs [1, 2]; it deploys FTTM practices 3.3, 3.4 and 3.5 directly [3]. It is the fastest route we know to the right product, and it takes weeks out of the schedule by finding errors before engineering converges rather than after beta.

What a council is

A council is a small standing group of actual users, assigned to one product and involved in the details of its definition, development and first season in the field. They are the "field designers": people who use the product on the job rather than experts who speak about it. Five per project gives enough diversity of view while keeping the group small enough to respond fast; three is the practical floor for the weekly work, and the formal sign-off check needs six (see the translation test below).

Most products have two personas: the user who operates the product, and the buyer or manager who decides the purchase, standardizes across a fleet or team and cares about integration with back-office systems. Each council carries one buyer or manager alongside the users; FTTM practice 3.1 calls this co-developing with the customer's customer [3]. Where the product carries a software platform, a separate enterprise council of buyers is the purest form of that practice.

Each council has a named owner inside the core team, a product manager or domain specialist with time allocated to the role, because frequency of contact decides whether the council contributes or drifts away. FTTM practice 3.8 makes the owner a role held by a named person [3], and that person sits with the team that uses the council's answers. Internal SMEs stimulate the dialogue; marketing supports recruiting.

Where it works in the VOC process

Most VOC fails because it starts at the wrong step. Teams show customers a solution and hear "sure, looks fine," which validates what the team already believed and teaches nothing about the problem the customer is trying to solve [4]. The council works the first two steps and the last one. It states the wants, checks the translation into product requirements, and then confirms the product every week through development, at introduction and into the field (Figure 1).

The council builds on the front-end research, the interviews, focus groups, conjoint and pricing studies. It interprets and tests those findings in the design and development phases, where most companies have no standing customer voice, and it leaves willingness to pay with the pricing study. It also takes the trade-off calls, the how-much-is-enough decisions on things like screen size, battery life and weight, which is where a small standing group beats a survey (practice 3.4, Teach-Negotiate-Tell [3]).

The translation deliverable is a sign-off. For each of the top customer requirements, members confirm that the derived product requirement is a valuable reading of the need, that it would be superior at introduction, and that they would buy it with the other specs at parity. Practice 3.5 sets the formal check at six customers [3], so add beta candidates to the council to reach six for that check. Where requirements live in a traceability matrix, the deliverable is a signed row.

Figure 1. The four VOC steps [5] and where the council is engaged. The translation step is where most definition errors originate and where the council pays back first; the sign-off is its deliverable there.

Why it produces the right product

A council keeps the team at the wants step until the wants are understood, then holds it accountable at translation. Customer requirements are the problem; product requirements are the means. Confusing the two is how teams ship features nobody asked for and miss the ones that matter [4]. Because the council stays through development and into the field, refresh runs continuously, so value is checked at the time of introduction, when it counts [6]. Direct contact also removes an interrupt FTTM names in practice 2.7: marketing and sales limiting the team's access to the customer [3].

Why now

Get the councils into the projects now, while product definitions are being finalized, and keep them involved throughout development. They will tell the team very fast whether it is right or wrong, and that saves the time a wrong call costs later. The same members then accelerate beta testing and its feedback. By that stage, beta has to confirm what the team already knows; a beta that tells the team something it did not know has arrived too late [2].

For the next group of products, form the council earlier still, at feasibility entry. FTTM practice 2.1 manages the fuzzy front end as a real phase, and 3.3 wants leading customers involved during gestation [3]. A council formed while definitions are being finalized is late by that standard, though still worth doing.

What it is worth

FTTM practice 1.6 uses cost of delay to justify provisioning [3]. A product with $3M of first-year incremental revenue loses about $8,000 a day, or $58,000 a week, straight-lined over 365 days. Cost-of-delayAI gives the proper figure; use incremental sales, not total product line sales. A Slack subscription and a few owner-hours a week are small against one week saved.

Plan around the season

Practice 3.13 includes when customers want the product, and many markets are seasonal [3]. Member availability drops in the busy season and beta timing depends on it, so set the meeting cadence and the beta windows around the season.

How councils accelerate the schedule

The schedule gain comes from five mechanisms, each one a known source of delay on technology programs.

Mechanism	What the council changes	Schedule effect
Earlier convergence	The what, who, why and value questions get answered by users in the first weeks instead of drifting to keep options open [6].	Fast teams diverge for a third of the program and converge for two thirds. A late turn toward convergence delays the launch, because convergence still needs its two thirds [6].
Error found before build	Members tell the team within days whether a definition or a prototype is right or wrong.	An error caught in definition is a conversation. The same error caught in beta is rework, and by then it is too late to fix cheaply [2].
Scope discipline	Users rank needs, so features that solve no field problem drop out. On one lateralworks program a set of wireless and software features left the first release this way [1].	Fewer features to build, test and qualify. Deferred features become the roadmap for the next release.
Shorter beta	Council members become the beta sites, as the four VOC customers did on a lateralworks GreenTech program [1].	Beta confirms what the team already knows instead of discovering it, so it runs shorter with fewer derisking loops [2].
Decision velocity	When a product debate stalls on competing expert opinions, a post to the council channel returns field evidence the same day. Position power leaves the room [1].	Decisions move at the under-24-hour turnaround FTTM practice 1.5 asks for, on the customer input practice 1.2 says should outweigh internal SMEs [3].

Table 1. Five schedule mechanisms. Quantify the total with Cost-of-delayAI using incremental sales, not total product line sales, for the daily figure.

Where it sits in the program

FTTM practice 3.3 names eight modes of customer involvement across a program [3]. The council covers requirements through the first field season. Two modes stay outside it: gestation, where the next wave of councils should start, and overrun, where the council becomes a mutual problem-solving forum if the program needs one. Connected products raise the stakes on the field season: software keeps releasing after the hardware ships, each release builds on the last, and a council that disbands at launch loses the people best placed to judge the next release (Figure 2).

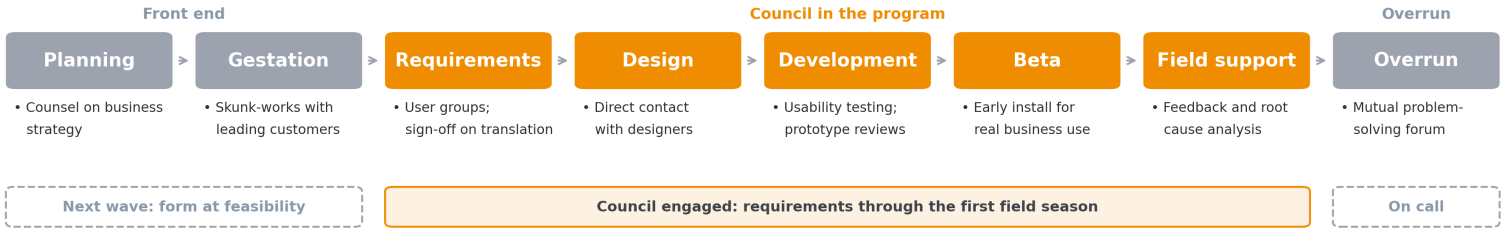


Figure 2. The eight VOC modes of FTTM practice 3.3 [3] and the council's span. Field support is in the remit; overrun is on call.

How to make it happen

1. Buy a corporate Slack subscription. Private channels and direct messages give control over who sees what.
2. Assign the owner inside each core team and give the role time. lateralworks supports that person; setup takes about twenty minutes.
3. Make the NDA step zero of onboarding, and write a one-line rule on what does not go into council channels. The most sensitive differentiating IP stays with inside people (practice 3.9).
4. Define one council of three to five users plus one buyer or manager, and one channel per product. Combine or separate as the products dictate.
5. Put the council touchpoints into the project schedule as dated tasks: requirement sign-off, prototype reviews, usability sessions, beta start and the first-season review. Scrub them weekly with the rest of the schedule and take open questions to the decision forum (practices 3.3 and 3.10).
6. Tell members where the commit points are, so they know when their input shapes this release and when it flows to the next.
7. Invite the team members who will work with the customers. Expect daily posts and same-day answers.
8. Onboard the first council, usually one already forming around a project, and start the weekly one-hour meetup. Set the cadence and the beta windows around the busy season.
9. Repeat for the next group of products, recruiting from current customer engagements, and form those councils at feasibility entry.
10. Scan the channels weekly with the Claude connector to Slack and consolidate the themes and open questions across councils.

Change control

Practice 3.6 warns that the more customers you ask, the more specification changes and rework you get, and it prescribes flexible containment: front-load changes, separate early from late commits, and run a change control board that convenes at once [3]. A weekly council feeding a channel generates change, so the commit points are part of the council's brief. Before a commit, council input shapes the definition. After it, input goes to the change board or to the next release; deferred features become the roadmap.

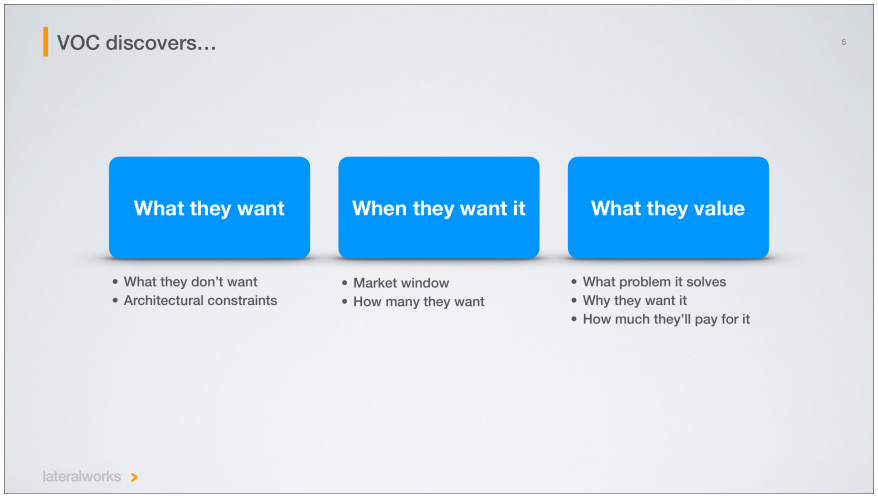
Keep it honest

Two things go wrong with councils. Members who lose contact stop contributing, so the owner role has time allocated to it. And the council must stay on problems and needs; the moment it becomes a feature show-and-tell it turns back into feature-based VOC and the schedule benefit disappears [4]. Both are visible if measured (practice 2.9 [3]). Report a few per-council indicators by exception in the weekly program scorecard: median hours to answer a post, meeting attendance, questions open longer than a week, requirement rows signed off, and decisions settled by council evidence.

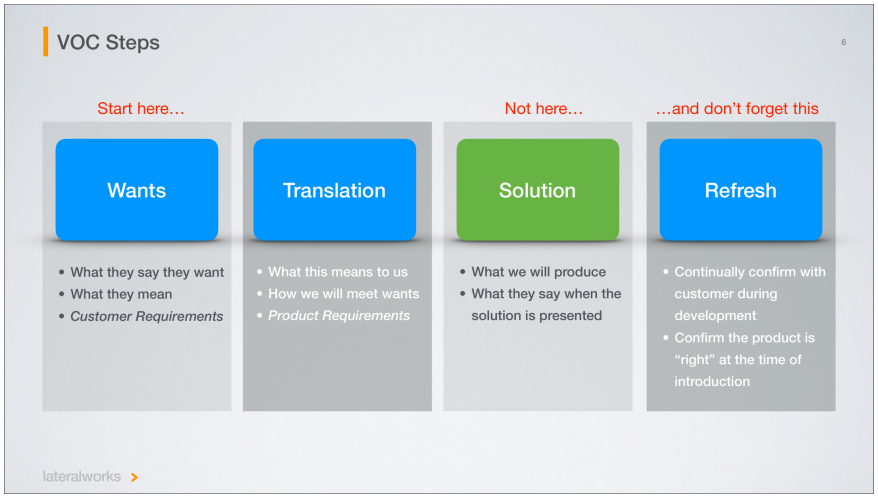
Appendix A

VOC training slides

Two slides from the lateralworks FTTM training module on voice of the customer, reproduced without change [5].



Slide 5. VOC discovers what customers want, when they want it and what they value.



Slide 6. The four VOC steps: start at wants, not at the solution, and keep refreshing.

Appendix B

References

Numbered in order of first citation. Items 1, 4, 6 and 7 are lateralworks.com articles; 2, 3 and 5 are lateralworks engagement, methodology and training material.

[1] Mitchell, N. "From target market to product definition using VOC." lateralworks, April 2019. https://lateralworks.com/ideas/from-target-market-to-product-definition-using-voc

[2] lateralworks. "Customer quick response councils." Practice note from a client engagement, September 2026.

[3] lateralworks. *FTTM New Product Development Best Practices.* Revision 2018.002. Practices cited: 1.2, 1.5, 1.6, 2.1, 2.7, 2.9, 3.1, 3.3, 3.4, 3.5, 3.6, 3.8, 3.9, 3.10, 3.13.

[4] Mitchell, N. "Feature-based VOC vs discovering customer needs." lateralworks, June 2019. https://lateralworks.com/ideas/feature-based-voc-vs-discovering-customer-needs

[5] lateralworks. "VOC discovers" and "VOC steps." FTTM training slides 5 and 6, VOC module.

[6] Mitchell, N. "What, Who, Why, and Value?" lateralworks, July 2019. https://lateralworks.com/ideas/what-who-why-and-value

[7] lateralworks. "FTTM System." January 2019. https://lateralworks.com/ideas/fttm-system