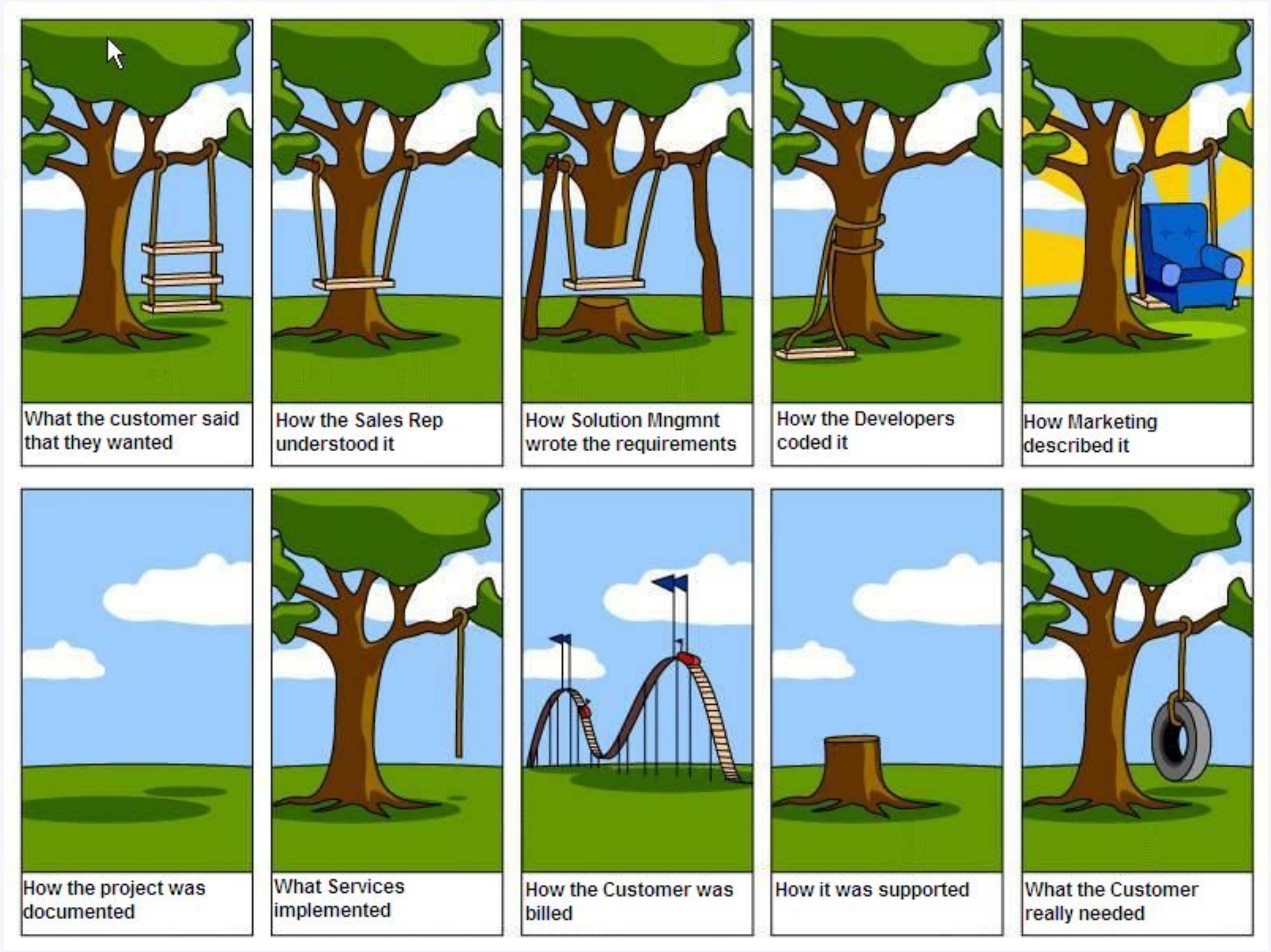

What the customer said that they wanted
How the Sales Rep understood it
How Solution Mngmnt wrote the requirements
How the Developers coded it
How Marketing described it
How the project was documented
What Services implemented
How the Customer was billed
How it was supported
What the Customer really needed

Goal-Setting in Complex Orgs

Ask a PM to help or do the PM’ing yourself

Nothing can control for luck, risk, etc.

Use processes that more consistently deliver customer value amid complexity.

We’ll skim non-customer goals today for simplification.

These processes are most important in large orgs.

About me

- Graduated from UW CSE 2014
- Took & TA'd CSE 403 with Prof. Ernst
- Current lead product manager for Pixel Camera (10 years)
- Mentored or managed 30+ PMs from 0 to 10 years of experience

Isaac Reynolds

Group Product Manager (GPM)

Table of contents

Product Goals

Successful businesses always start from the user's goal

Marshall Field's

The customer is always right.

Google

Focus on the user and all else will follow.

amazon

customer obsession rather than competitor focus.

We will truly understand [customers'] needs.

Then we work through all our other stakeholders’ goals

Engage often

Our Eng Team	Develop the solution rapidly with minimal cost.
Our Company	Drive revenue, profit, awareness, brand attributes, etc.
SysAdmin / IT	Easy setup, operation, and replacement.
Customers	Competitive pricing.
Users	Solve problems without creating too many new ones.
Platform	Target the newest platform and/or device features.
Regulators	Maintain a fair market that’s safe for consumers.

Engage less often (More surprises, more risks

You can separate all these opinions into a variety of types of goal

User Goals

The user is spending time in your product because they *want* something. What is it? It might be a feeling, task, creation, etc.

Business Goals

Business goals are among the most complex and extensive. You will have to consider brand, regulation, finances, timeline, and more.

Google Ex.: "... durably re-engineer our cost base."

Product Vision

Your vision is brief – just 1 sentence that bounds the *category* of problem you're trying to solve and what *aspects* of it you're willing to work on. It is applied strictly to exclude all kinds of projects.

Google Ex.: "Organize the world's information and make it universally accessible and useful"

Product Principles

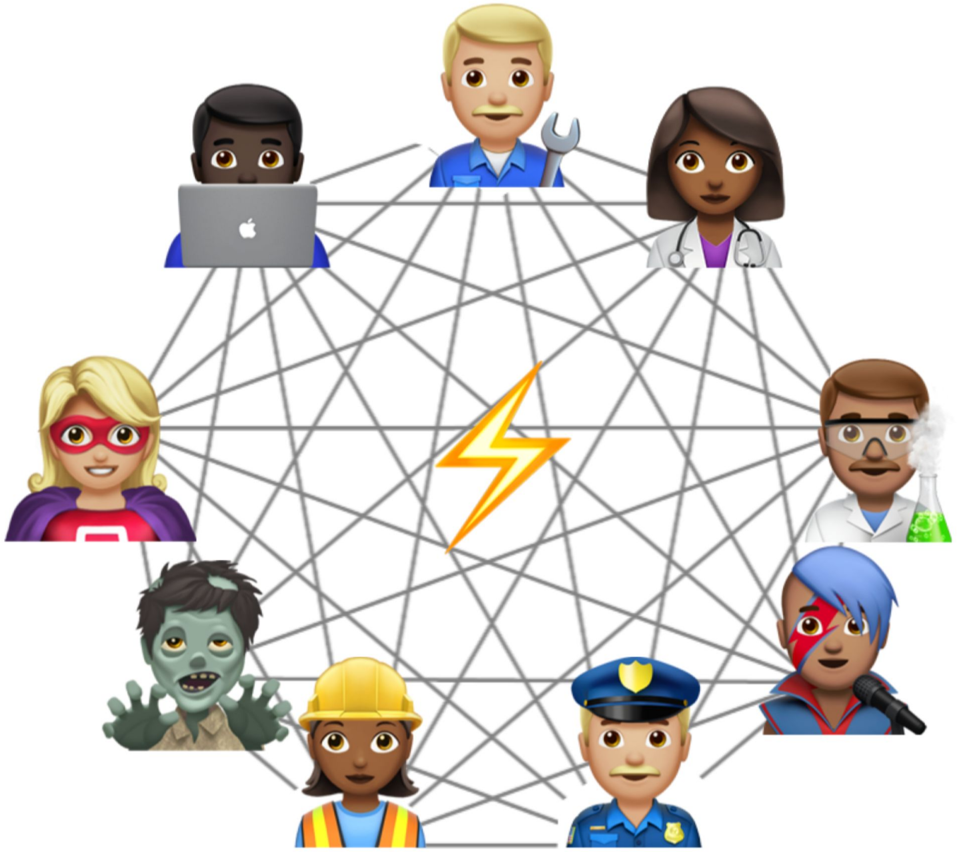
Two of us leveraged their skill working on ambiguous, complex problems to build compelling business cases and resolve conflicts.

Google Ex.: "Focus on the user and all else will follow"

Product Requirements

Precisely describe the product's functionality in terms understandable by your engineers and customers, with as close to zero ambiguity as possible.

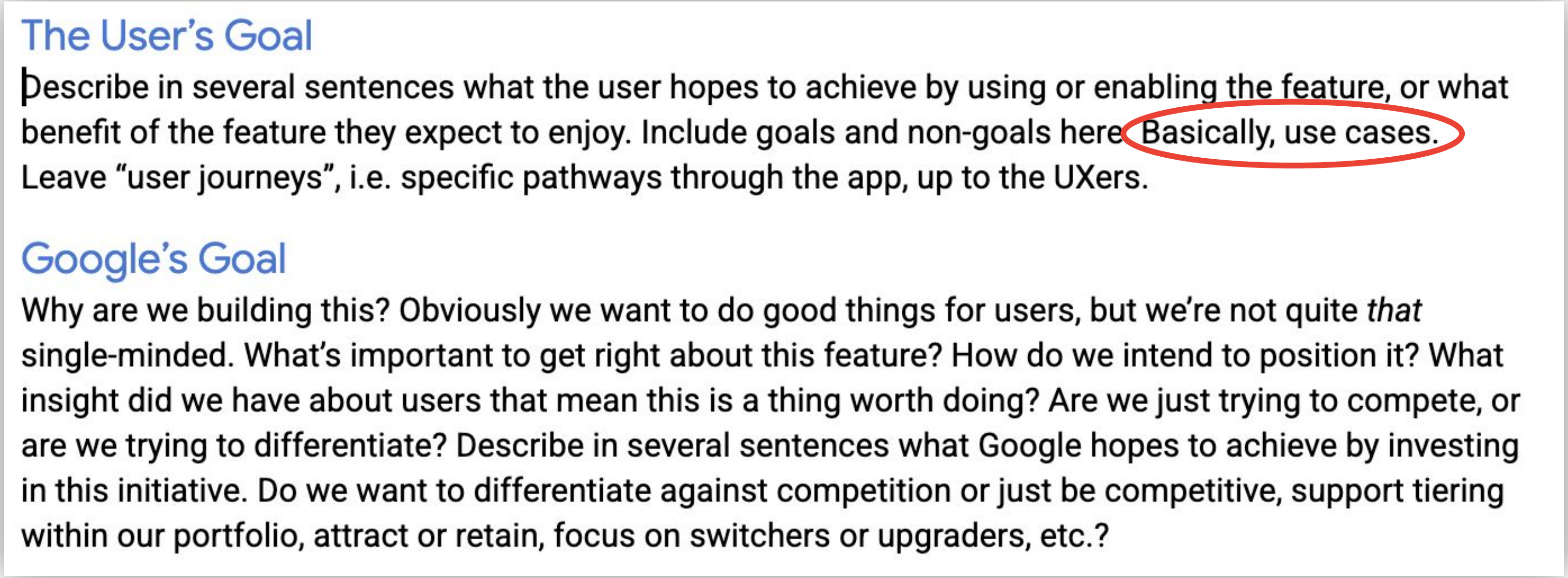
Goals should be documented to focus debate and save outcomes.



If one of those people is off-base, they'll cause their collaborators to be off-base, too!

Google

Put goals at the beginning of all your Product Requirements Documents (PRDs)



The User's Goal

Describe in several sentences what the user hopes to achieve by using or enabling the feature, or what benefit of the feature they expect to enjoy. Include goals and non-goals here. Basically, use cases. Leave "user journeys", i.e. specific pathways through the app, up to the UXers.

Google's Goal

Why are we building this? Obviously we want to do good things for users, but we're not quite *that* single-minded. What's important to get right about this feature? How do we intend to position it? What insight did we have about users that mean this is a thing worth doing? Are we just trying to compete, or are we trying to differentiate? Describe in several sentences what Google hopes to achieve by investing in this initiative. Do we want to differentiate against competition or just be competitive, support tiering within our portfolio, attract or retain, focus on switchers or upgraders, etc.?

Use the “use case” format.

As a [user], I want to [action] so that [result]

As a parent, I want to take sharp photos of my kids in medium-low light so I can have memories of early holiday mornings.

As a creative, I want to adjust the look-and-feel of my photos so that they match how I remember the moment.

As a photojournalist, I want to be able to prove where and when my pictures were taken so that I can defend their authenticity on the global stage.

As a YouTube Shorts creator, I want to caption my videos so that people without sound don’t skip my videos.

As a restaurateur, I want to take fresh, juicy-looking photos of food so that customers want to eat at my restaurant.

Or the "user journey" format.

A sequence of actions taken by the "system" and the "actor"

This is almost a use case …

Actor: As a parent ("actor"),

Goal: I want to take pictures of my young kids where they're all smiling.

Flow:
1. User arranges the family for a photo.
2. User presses the shutter 3 times in 5 seconds.
3. System saves 3 full-quality images.
4. User taps "gallery".
5. System opens the gallery and shows a button to "select a better moment".
6. User taps the button.
7. System automatically creates and shows the "Best Take".
8. User taps "Save as copy".*
9. System shows the user the saved copy in the gallery.

* Of course, the user might also tap "cancel". You could represent all of a product's user journeys as a tree, where each node is a new UI and each branch is a user decision. Documenting this tree of overlapping user journeys efficiently can improve communication.

Goals should be broadcast so everyone can work toward them independently.

Make goals SMART whenever you can

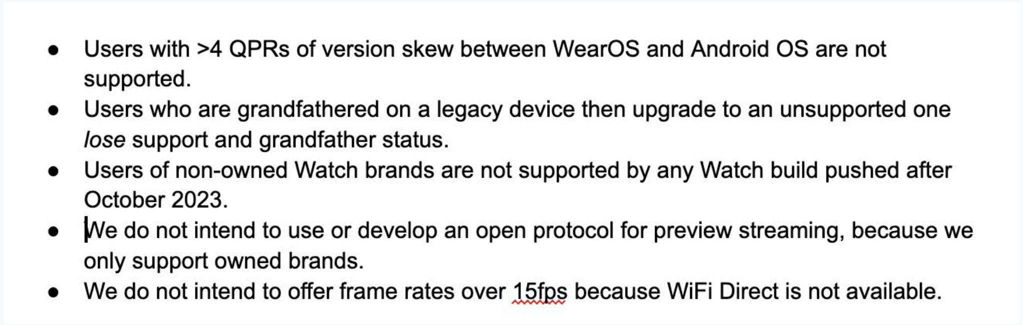
Specific	Precise word choice, clear, complete, and unambiguous. Addresses all key questions. E.g., "the user can enable a processing pipeline that's 50MP end-to-end from sensor mode to JPEG".
Measurable	A clear way to say "we're done" or at least "we're heading the right direction". We'll talk about common *quantitative metrics* later today. E.g., "the app will show a live preview with 3A converged within 500ms in 80% of launches."
Achievable	Appropriately "scoped" to the time, people, skills, technologies, etc. available to you. This might be a pretty small goal!
Relevant	You're always working in a larger context, like your boss's goals, or your company's. Your goal should help achieve those larger ones.
Time-Bound	A time when you'll check back on your *measure* to see how you're doing. E.g., "The initial product launch will include GAN-based upcaling".

Google

Excluded goals are just as important as included ones

- Problems you're not solving for the user.
- Features that won't be available.
- Stakeholders' goals you won't be able to satisfy.
- Data you won't collect.
- Competitive gaps you won't satisfy.
- Anything you discussed in depth and decided *against*.

Every Pixel Camera requirements document ends with our non-goals.



- Users with >4 QPRs of version skew between WearOS and Android OS are not supported.
- Users who are grandfathered on a legacy device then upgrade to an unsupported one *lose* support and grandfather status.
- Users of non-owned Watch brands are not supported by any Watch build pushed after October 2023.
- We do not intend to use or develop an open protocol for preview streaming, because we only support owned brands.
- We do not intend to offer frame rates over 15fps because WiFi Direct is not available.

Would this have been a good initial goal for Uber?

~~“Make it easy to get a ride.”~~

This is a poor goal.

What are we going to build to make it easier? How will we know we’ve made it easier? How much easier is enough? How much time can we afford to spend trying?

It’s important to be specific!

What might be a better goal for Uber?

“Give infrequent taxi riders certainty about their ride”

This would lead to the features that helped Uber transform the industry.

- That a car will be available. (“Gig” economy with dynamic pricing)
- When they’ll get picked up. (GPS tracking)
- Whether it’s safe. (Driver background checks)
- When they’ll get home. (GPS navigation)
- What they’ll pay. (Cost calculator)

Google

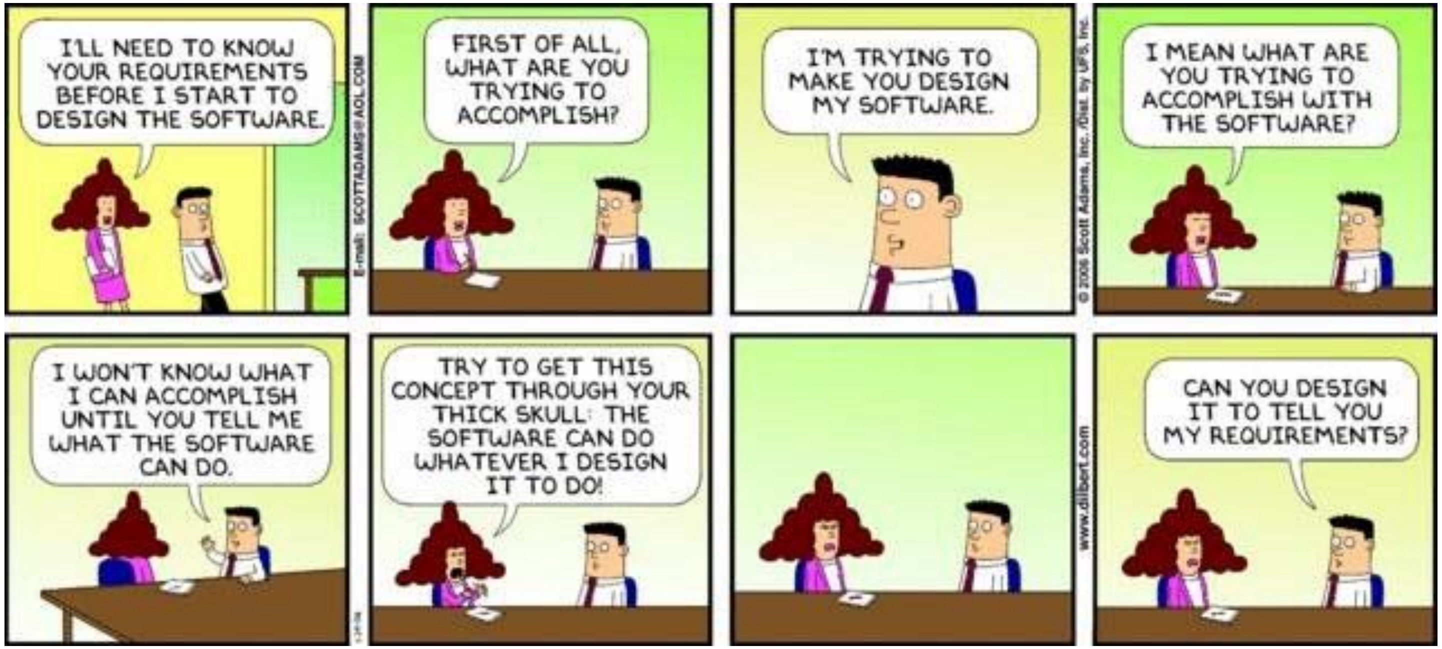
Requirements

Product requirements are a more-specific way to express goals.

No matter your product or process, always create requirements.

- Used in all development philosophies (Waterfall, Agile, Prototyping, Spiral, eXtreme Programming, etc.).
- Especially valuable in long-lead waterfall-types or client-specific contractual requirements.

When should you do requirements?


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
E-mail: SCOTTADAMS@AOL.COM
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com

Google

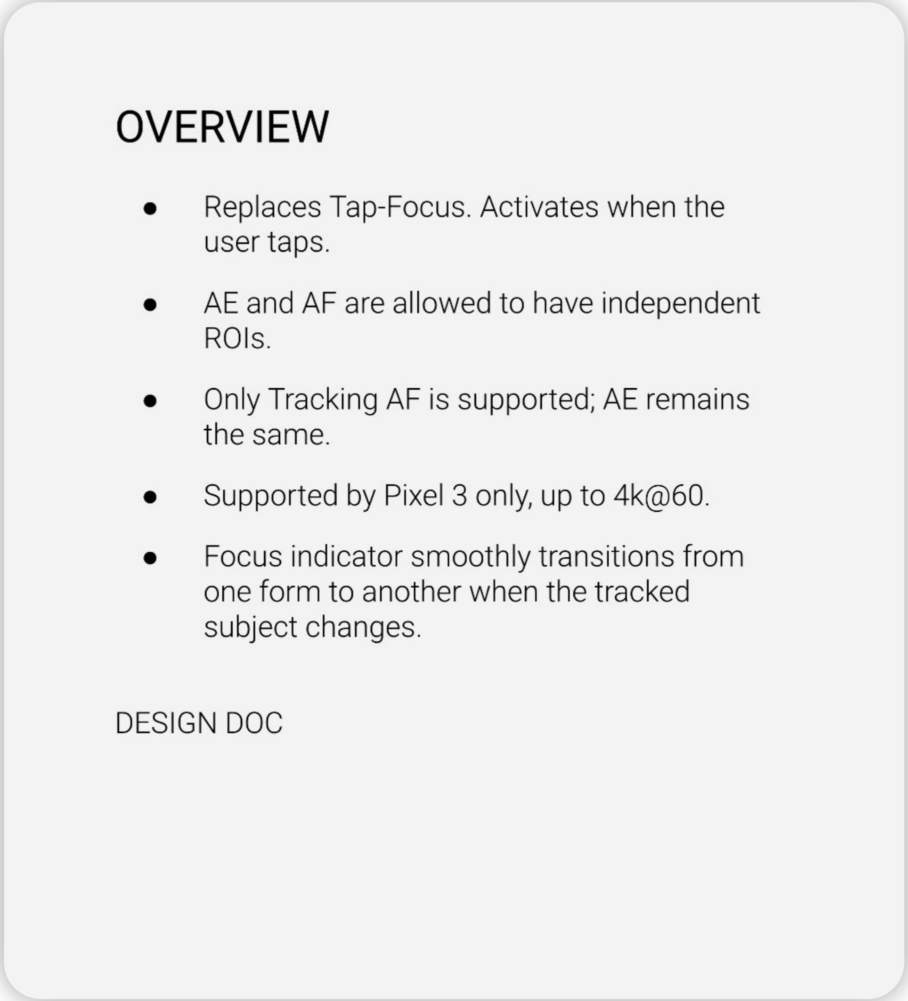
Elicit requirements by connecting with your user

- Be a user yourself (but a double-edged sword!).
- Talk with users informally (hallway chats, mixers)
- Talk with users formally (interviews, surveys, diary studies, field studies, forced ranking)*. Consider quantitative and qualitative options.
- Build low-fidelity prototypes (mocks, UX prototypes, eng prototypes).
- Launch and get feedback early ("launch and iterate").

* Results are complex enough to merit an entire specialty: user experience research (UXR).

Write them carefully.

- Describe user-visible attributes.
- Represent collaborators' key goals.
- Leave UX / eng as unconstrained as possible.
- Specify constraints rather than solutions.
- Avoid rigid templates / formats.
- Avoid "and" (break it into two if you want "and").
- Quantify the goal whenever possible. Think SMART!
- Assign priorities.
- Clarify dependencies.



OVERVIEW

- Replaces Tap-Focus. Activates when the user taps.
- AE and AF are allowed to have independent ROIs.
- Only Tracking AF is supported; AE remains the same.
- Supported by Pixel 3 only, up to 4k@60.
- Focus indicator smoothly transitions from one form to another when the tracked subject changes.

DESIGN DOC

A “notification” (not necessarily Android notification) should activate as soon as the triggering criteria are met:

- *P0* Haptic components that are likely to be felt from [redacted].
- *P0* Aural components that are likely to be heard from [redacted].
- *P0* If the user notices the notification, they can override it manually, which prevents it from reappearing for some reasonable period thereafter.
- If the user does not notice the notification or takes no action
  - *P0* The [redacted] is automatically disabled.
  - *P1* The user can find a history that a previously-issued notification caused the [redacted] to be disabled.
- *P0* If the [redacted] will be automatically disabled, it must be turned off within 30 seconds of the start of the last trigger condition to be met (e.g., if it took 3 seconds to meet the threshold, then you have 30 - 3 = 27 seconds before the [redacted] must be automatically turned off).
- *P1* The “notification” overrides current notification settings (e.g., DnD, vibrate / silent).
- *P2* The user can disable all future notification.

The operation of the feature is logged.

- For each [redacted] session:
  - *P0* Average distance measured by [redacted].
  - *P1* Variance in distance measured by [redacted].
  - *P0* P99 (near-maximum) distance measured by [redacted].
  - *P1* P01 (near-minimum) distance measured by [redacted].
  - *P0* Duration.
  - *P2* Source of activation (see list below).
  - *P0* Whether the notification triggered.
  - *P0* Whether the auto-off was overridden. (False Positive)
  - *P0* Whether the [redacted] was reactivated within 10 seconds of being automatically disabled. (False Positive)
- *P1* The number of customers who reached out to customer support [redacted]. (False Negative)
- *P1* The number of customers who reached out to customer support with a complaint about the auto-off. (False Positive)

In this case, the “user” is a member of the business. We want to document everything the engineering team has to build.

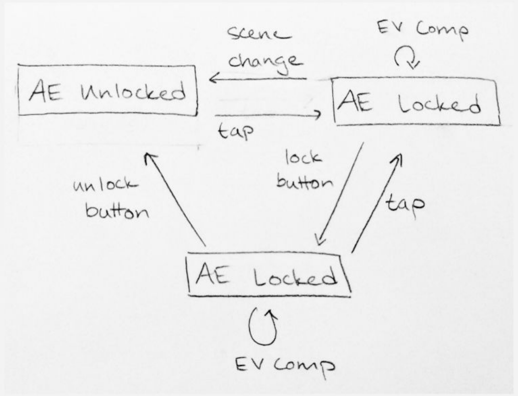
Google

Don’t be dogmatic; pick the most-effective format.

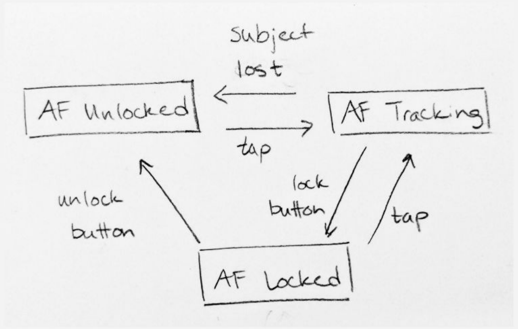
PHOTO MODES

- All photo-oriented modes (e.g., incl. Portrait)
- AE is unchanged.
- AF adds tracking.

AUTOEXPOSURE



AUTOFOCUS



Consider your audience

Every specialist will use your requirements for their own purpose – make sure your requirements are useful to all your stakeholders.

Consider the example of some signed metadata in a JPEG image …

“The metadata shall be signed in the TEE.”

Engineering hears: Processing has to *fit* in the constrained TEE.

Marketing hears: The feature supports a “security and privacy” narrative.

User hears: The metadata is trustworthy.

Customer Support hears: If the feature isn’t working, the user’s device may be compromised.

Test / QA hears: Penetration testing / red-teaming should be conducted.

Document everything!

- User features
- Performance and System Health
- Reliability
- Scalability
- Warranties or maintenance goals
- Possible or likely future goals
- Target platforms or environments
- Regulatory and legal
- External documentation, user “help”
- Marketing claims
- Logging and success metrics
- Manual testing guides
- Accessibility
- Internationalization, localization, language support
- Troubleshooting guides
- Leak prevention
- Threat models and security guarantees
- User privacy
- Simplicity and usability

Document Everything! (Examples)

- [Features] A short video is captured of ±2sec around each shutter press.
- [Performance] The short video will start playing within 200ms of opening the image in 99 percent of cases.
- [System Health] The short video won’t not exceed 2MB in any case.
- [Reliability] The video loss tolerance is 100 ppm of times the trigger conditions are met.
- [Scalability] The short video will sustain 5 shutter presses per second for 5 seconds on a “clean” device.
- [Warranties, maintenance] The short video feature will not be removed from any device that included it at launch.
- [Possible future goals] The short video may someday share storage space with the primary image using temporal compression formats.
- [Target environments] The short video will run on Pixel 3+.

Possible requirements for a feature like Motion Photos or Live Photos

Document Everything! (Examples)

- [Regulatory] The short video will be subject to only non-biometric algorithmic processing.
- [External documentation] The public Help Center will describe the short video feature.
- [Marketing] The short video feature will be globally available on launch day.
- [Logging and dashboards] Add a field to per-image logs including whether the condition was triggered and whether the short video was saved.
- [Accessibility] The aural description of the on-off toggle will be “short videos will be captured; tap to turn off” when turned “on”.
- [Localization & Internationalization] The feature will be available in all device launch markets (link).
- [Troubleshooting] All triggering conditions are documented for customer support and this document is updated once per release.

Possible requirements for a feature like Motion Photos or Live Photos

Document Everything! (Examples)

- [Leak prevention] The phrase “short video” will be added to the binary scanner’s database.
- [Security guarantees] The short video will be processed exclusively by the HW and ISP, not SW.
- [Privacy guarantees] The user can turn off capturing short videos.
- [Simplicity] The short video will be captured automatically using the existing shutter button press.

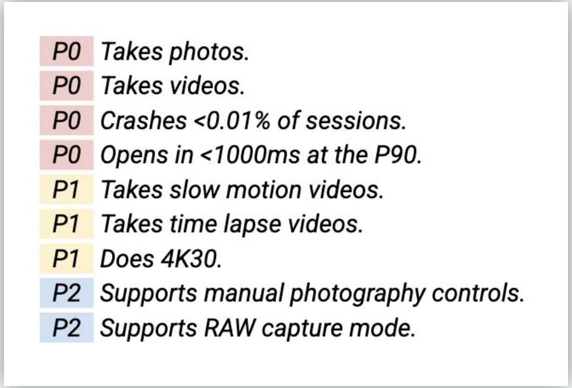
Possible requirements for a feature like Motion Photos or Live Photos

Prioritize ruthlessly

If everything is a "Priority 0" (P0), then nothing is!

- P0 means we'd be embarrassed not to have this.
- P1 is what makes the feature better than the competition.
- P2 is nice to have.

Consider the example of a simple "camera" app.



P0	*Takes photos.*
P0	*Takes videos.*
P0	*Crashes <0.01% of sessions.*
P0	*Opens in <1000ms at the P90.*
P1	*Takes slow motion videos.*
P1	*Takes time lapse videos.*
P1	*Does 4K30.*
P2	*Supports manual photography controls.*
P2	*Supports RAW capture mode.*

Upgrade to stack ranks ………… Or the points system

Use a stack rank!

1. Your first priority.
2. Your second.
3. Your third.
4. And so on …

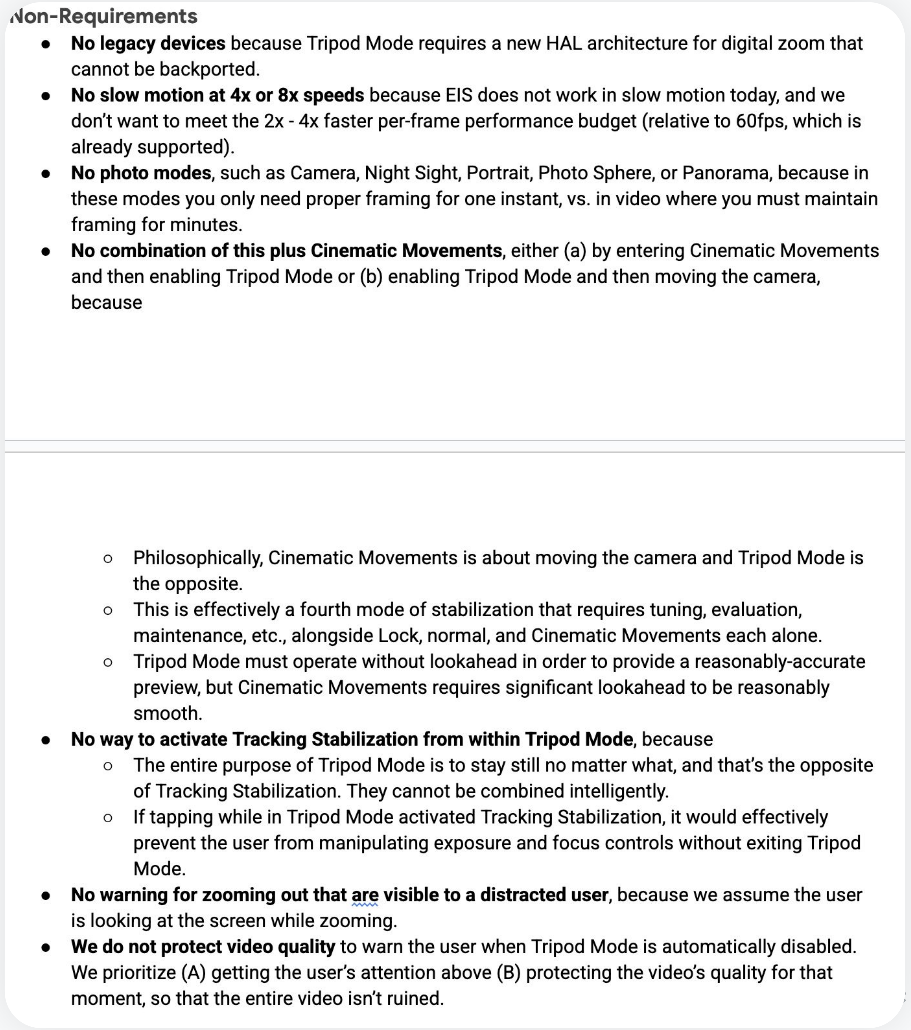
We have 100 points to assign.

- 30: Your top priority.
- 25: Your close second priority.
- 5: A a distant third priority.
- 2: Something you care less about.
- 2: Something else you care less about, but equally less.

Avoid “feature creep”.

What is it?	Requirements are constantly added, which tends to delay release past expectations. That damages morale and may violate contract obligations.
Why does it happen?	Developers like building features. Marketing and sales like to brag. Users always want more. Executives can easily see progress.
How do I avoid it?	Make timeline a requirement (SMART). Push engineers to make accurate resource estimates. And set arbitrary targets (e.g. only 20% of these ideas can be “yes”).

Document the requirements you're excluding!



Non-Requirements

- **No legacy devices** because Tripod Mode requires a new HAL architecture for digital zoom that cannot be backported.
- **No slow motion at 4x or 8x speeds** because EIS does not work in slow motion today, and we don't want to meet the 2x - 4x faster per-frame performance budget (relative to 60fps, which is already supported).
- **No photo modes**, such as Camera, Night Sight, Portrait, Photo Sphere, or Panorama, because in these modes you only need proper framing for one instant, vs. in video where you must maintain framing for minutes.
- **No combination of this plus Cinematic Movements**, either (a) by entering Cinematic Movements and then enabling Tripod Mode or (b) enabling Tripod Mode and then moving the camera, because
    - Philosophically, Cinematic Movements is about moving the camera and Tripod Mode is the opposite.
    - This is effectively a fourth mode of stabilization that requires tuning, evaluation, maintenance, etc., alongside Lock, normal, and Cinematic Movements each alone.
    - Tripod Mode must operate without lookahead in order to provide a reasonably-accurate preview, but Cinematic Movements requires significant lookahead to be reasonably smooth.
- **No way to activate Tracking Stabilization from within Tripod Mode**, because
    - The entire purpose of Tripod Mode is to stay still no matter what, and that's the opposite of Tracking Stabilization. They cannot be combined intelligently.
    - If tapping while in Tripod Mode activated Tracking Stabilization, it would effectively prevent the user from manipulating exposure and focus controls without exiting Tripod Mode.
- **No warning for zooming out that are visible to a distracted user**, because we assume the user is looking at the screen while zooming.
- **We do not protect video quality** to warn the user when Tripod Mode is automatically disabled. We prioritize (A) getting the user's attention above (B) protecting the video's quality for that moment, so that the entire video isn't ruined.

A little quiz …

Good or bad requirements? (and why?)

- The system will enforce 6.5% sales tax on Washington purchases.
- The system shall display the elapsed time for the car to make one circuit around the track within 5 seconds, in hh:mm:ss format.
- The product will never crash. It will also be secure against hacks.
- The server backend will be written using PHP or Ruby on Rails.
- The system will support a large number of connections at once, and each user will not experience slowness or lag.
- The user can choose a document type from the drop-down list.

Good or bad requirements? (and why?)

- The system will enforce 6.5% sales tax on Washington purchases.
- The system shall display the elapsed time for the car to make one circuit around the track within 5 seconds, in hh:mm:ss format.
- The product will never crash. It will also be secure against hacks.
- The server backend will be written using PHP or Ruby on Rails.
- The system will support a large number of connections at once, and each user will not experience slowness or lag.
- The user can choose a document type from the drop-down list.

Google

A better version of the requirements above

- P0: The user can see their “lap time” in an obvious, apparent place.
- P0: The user can see the lap time with maximum precision available.
- P0: The “lap time” must be within 10ms of the true lap time.
- The system functions in the following environments:
  - P0: Unrooted devices with signed OS and top-1000 apps from Play Store.
  - P1: Rooted devices with signed OS and malicious sideloaded apps.
  - P1: Rooted devices modified OS and malicious sideloaded apps.
  - Non-Goal: Anyone with the ability to replace the camera sensor.
- P1: The user can make documents of the following types: PDFs, Word, JPEG.

WE'VE MADE PROGRESS. THE CUSTOMER'S REQUIREMENTS HAVE GONE FROM BEWILDERING TO VAGUE AND AMBITIOUS.
REQUIREMENTS MEETING TODAY 1:00
MODERNANALYST.COM

Google

Product Metrics

A “north star” metric is the one you hold above all others

Product metrics are the least ambiguous way to express goals.

And engineers love them some metrics, let me tell you.

Every project or product has a “north star” to focus on

- **Users**: The number of unique users who log into your product in the rolling 28 days (“28DA”).
- **Usage**: The number of times a particular feature is used in the last 7 days.
- **Retention**: The percentage of users in the last 30 days who were also users in the previous 30 days.
- **Awareness**: The number of users who can recall your brand aided or unaided.
- **Consideration**: The percentage of phone shoppers who evaluate your phone.
- **Impressions**: The number of people who saw your ad.
- **Conversions**: The percentage of people who clicked your ad that made a purchase within 30 days.
- **Latency**: The time from some start event to some end event, as measured by a wall clock (as opposed to a CPU clock, for example).

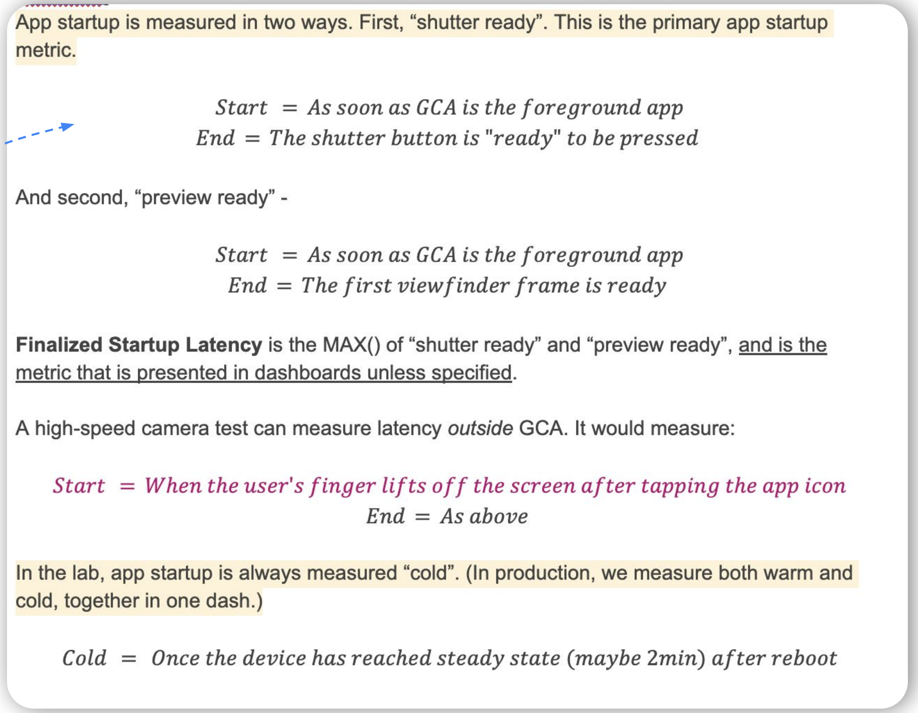
Every project or product has a “north star” to focus on

- **Stability**: The portion of tasks that are allowed to end in unhandled / unexpected failures.
- **ARPU**: The average revenue generated by each user, or the total revenue divided by total users.
- **Revenue**: Total dollars received by the company from customers.
- **Cost**: Total dollars spent sustaining the product, e.g. payroll, rent, servers, returns, etc.
- **Profit**: Total revenue less total costs, which we *hope* is a positive number.
- **Support contacts**: The percentage of customers that contact support with an issue.
- **Remorse returns**: The percentage of sales that are returned for refund.

You must define each metric extremely precisely

Consider there are seven different "app startup latency" distributions described here.

- Shutter-ready in-app, cold and mixed
- Preview-review in-app, cold and mixed
- Finalized mixed
- Shutter-ready by camera, cold
- Preview-ready by camera, cold



App startup is measured in two ways. First, "shutter ready". This is the primary app startup metric.

$$Start \ = As\ soon\ as\ GCA\ is\ the\ foreground\ app$$
$$End \ = The\ shutter\ button\ is\ "ready"\ to\ be\ pressed$$

And second, "preview ready" -

$$Start \ = As\ soon\ as\ GCA\ is\ the\ foreground\ app$$
$$End \ = The\ first\ viewfinder\ frame\ is\ ready$$

Finalized Startup Latency is the MAX() of "shutter ready" and "preview ready", and is the metric that is presented in dashboards unless specified.

A high-speed camera test can measure latency *outside* GCA. It would measure:

$$Start \ = When\ the\ user's\ finger\ lifts\ off\ the\ screen\ after\ tapping\ the\ app\ icon$$
$$End \ = As\ above$$

In the lab, app startup is always measured "cold". (In production, we measure both warm and cold, together in one dash.)

$$Cold \ = Once\ the\ device\ has\ reached\ steady\ state\ (maybe\ 2min)\ after\ reboot$$

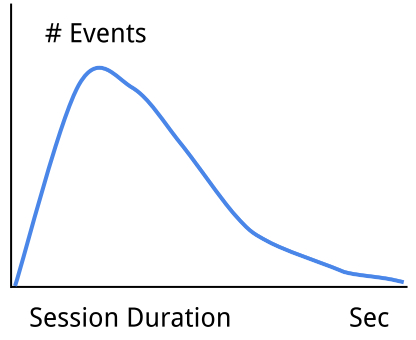
Most often statistics are distributions

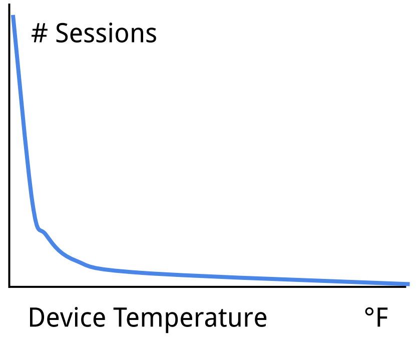
Colloquially, distributions may be:

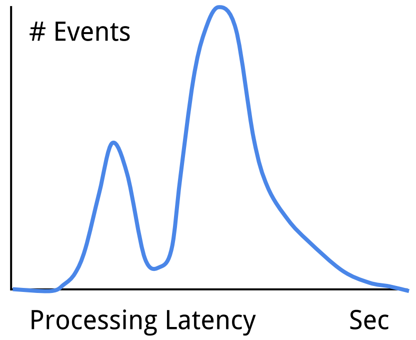
- Bimodal (or trimodal, etc.)
- Long-tail
- Gaussian
- Poisson
- Something else

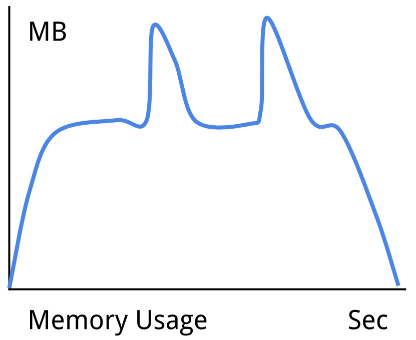
Summary statistics may be:

- Arithmetic mean (average)
- Median (P50)
- Geometric mean
- Mode
- Weighted mean
- P90, P95, P99 (long tail)
- % of events, devices, users, sessions
- Parts per million of the same
- Anything you like!





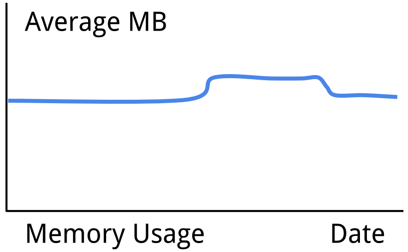




How would you summarize each of the distributions here as just one number?

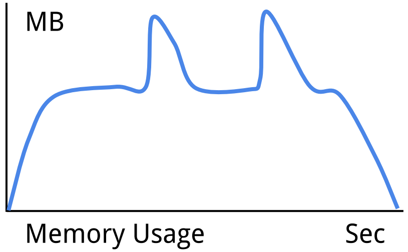
Consider, for example:

- Arithmetic mean (average)
- Median (P50)
- Geometric mean
- Mode
- Weighted mean
- P90, P95, P99
- % of events
- Parts per million (ppm)
- Anything you like!



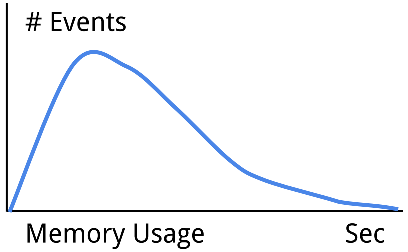
I want to see typical memory usage in lab tests on the latest internal builds. When memory increases, there may be a regression.

±Δ MB vs. last week



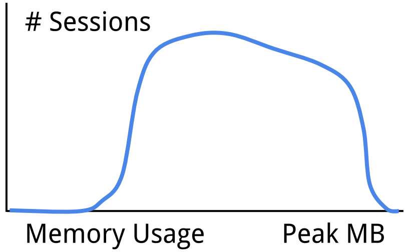
I want to see memory usage over time for a single app session. When memory spikes, it may evict background apps.

Peak MB



I want to see how long it takes to allocate memory on initial application launch. When the tail is long, preview drops frames.

P99 Seconds

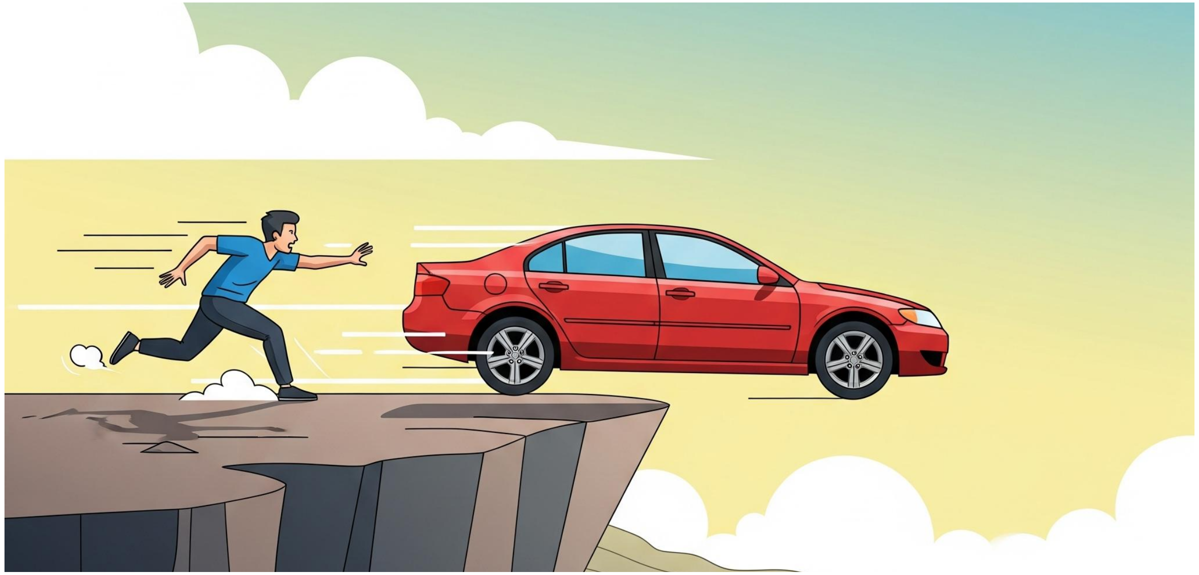


I want to see the peak memory used by sessions in public. When the tail is tall past our budget, we're evicting background apps too often.

PPM > Budget

Google

Metrics allow engineers to create requirements independently (wow!), but it’s a little like putting a brick on the gas pedal – make sure the car is pointed the right way first!



Setting targets is very tricky

- **User-driven**: If you want to build a *really good* product, you might ask users directly what would make a difference.
- **Competitive**: If you want to make a product that's *good enough*, you might simply choose to equal or better the alternatives.
- **Iterative**: Choose relative to your existing in-market product, especially if competitive isn't realistic or if you're the market leader.

Your target might be a **launch blocker** or a **goal.** You should specify both!

- **Launch blocker**: a "P0", i.e. the product doesn't ship until this goal. Often set at "no regression".
- **Goal**: A "P1", i.e. what you think is achievable with the resourced projects.

Measuring low-volume events requires deep understanding of your data.

Events measured in 10s of parts per million are affected by:

- People pretending their non-Pixel device is a Pixel.
- People who updated their app but not OS continue sending bugs reflecting now-resolved OS-level logging bugs.
- Camera automated tests run by a carrier partner report many events.
- Kiosks in building lobbies that run 24/7 even overnight.
- Device / platform crashes preventing reporting.
- Devices with spotty connectivity reporting events intermittently.
- Race conditions that report events in the wrong order.
- Devices run out of battery and report events hours apart that should be reported seconds apart.

Tricks of the trade™

Imagine a feature that sends the user input to a server for processing and then shows the user the result. Your product terminates the user journey with a timeout error if it hasn't received a response in 5 seconds.

Your metric is what portion of requests terminate with a timeout error. It is 5%.

What portion of events would end in success if you adjusted the timeout to 6 seconds?

What is the easiest way to move the metric toward "better"?

If the user closes the tab after 4 seconds and the request hasn't returned, does your metric call this "success", "failure", or neither?

Can you think of a product you've used whose timeout is obviously too long? Wanna bet what might've happened?

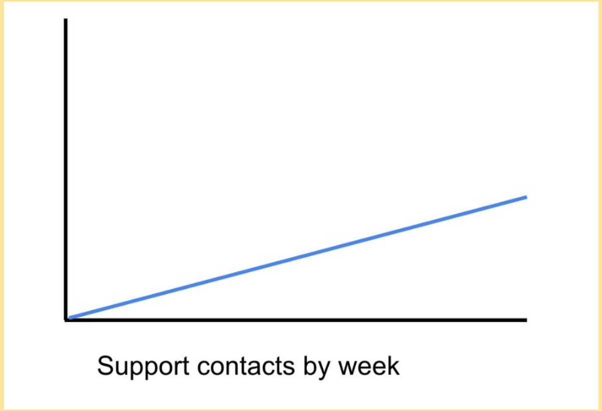
Google

Measuring the absence of an event is always very tricky

Tricks of the trade™

You sell a product along with a customer support contract. Users are free to call your support phone number if they have an issue.

Your metric is the number of support contacts per week. It looks like



What would you do, if anything, if you managed the support team?

What would you do, if anything, if you managed the product feature roadmap?

Are users more happy with your product, less, or something else?

Does your answer change if you're selling more licenses every week than the one before?

Choose a reasonable normalization for your data

Tricks of the trade™

Your camera product has 2 features: a JPEG compression algorithm required to save still images and also a panorama feature.

Your metric is what portion of users used each feature in the last 7 days ("%7DAU"). It's 90% and 0.05%, respectively.

Which feature are users more aware of?

Should you remove the Panorama mode?

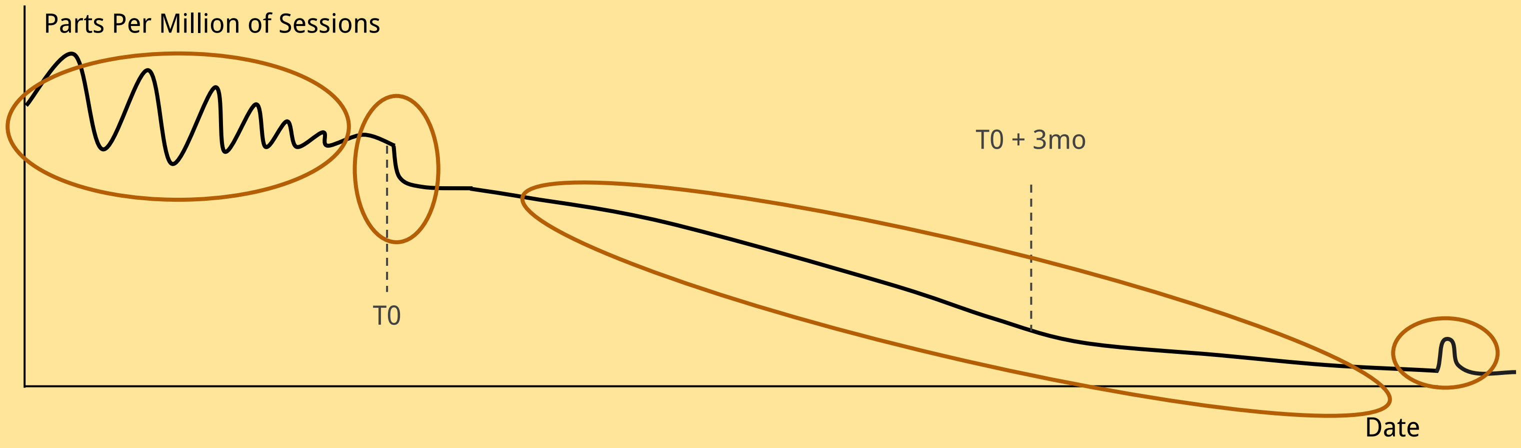
How much effort should your team invest in the future of JPEG encoding vs. Panorama?

What do you do when your boss asks you to report usage for all features to drive prioritization?

Don’t make unfair comparisons, and consider balance metrics

Tricks of the trade™

You’re building Pixel Cameras. Your video recording mode uses 3W continuously, much of which is converted to heat by the CPU and dissipated into the device chassis. You measure what portion of sessions heat the device chassis to at least 50C. For your newest phone, it looks like …



Tricks of the trade™

You're building an LLM-based chatbot. A team of human raters test against a consistent set of 2,000 inputs every day, evaluate the outputs, and publish a single score. Engineering teams identify regressions during development this way, and you approve launches using this metric.

You approve a launch based on a big improvement. But you find post-launch that most users find your update worse. What might be going on?

A story about normalization based on a job interview I ran with a senior PM candidate.

30% of what?

30% of eligible users.

Who's an eligible user?

Users who have previously engaged with a similar feature in the past.

What does it mean to "engage"?

They have to have used a similar feature a certain number of times in the past month.

How many times?

I chose 5 times in the past month.

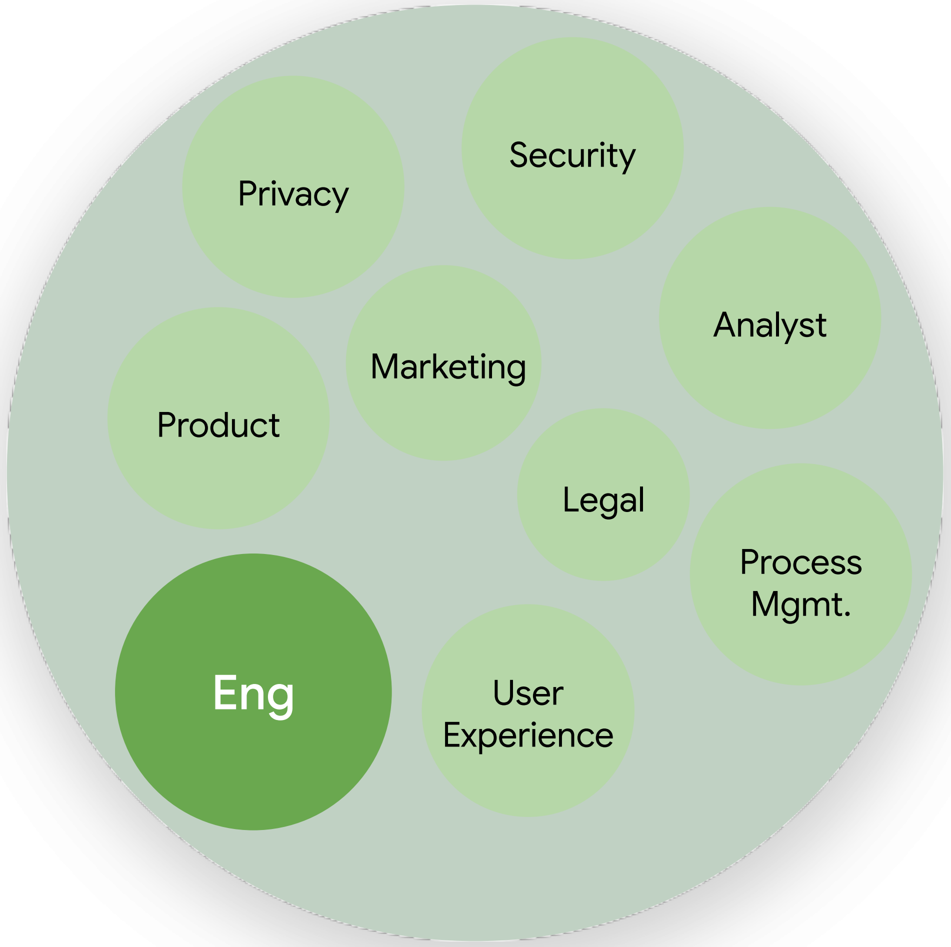
Why did you choose 5? Why not 6, or 7?

It seemed like a couple times a week, that feels like "a lot" and it gave us a 30% number that looked good at the exec review.

This was a no-hire.

Team Organization

Teams are made up of people with different specializations performing different functions



Every function has different skills

If you have this degree ...	You might work in ...	And day-to-day you'd ...
Computer Science, Electrical Engineering	Engineering, Process Mgmt., Product Mgmt.	Write detailed code that constitutes the product and its tooling, testing, release, and maintenance.
Drawing, Painting, Industrial Design – generally Art-related	User Experience	Draw out plans for what the product should look like and where every button, text, etc. goes.
Business Administration	Product Mgmt., Marketing	Determine what buttons and information the user needs in your product.
Juris Doctor	Legal	Confirm the product complies with relevant regulations.
Information Technology, Informatics	Privacy, Security Consultants, Process Mgmt.	Confirm the product complies with relevant corporate policies and advise on how to exceed them.
Applied Mathematics, Statistics, Finance	Strategy, Analyst, Product Mgmt.	Plan changes to the product based on how users use it and its competitors.

Today’s lesson is primarily the “product manager” function …

… Not the other types of “PM”!

Our focus today!

01 **Product Manager (“PM” or “PdM”)**

Creates roadmaps that drive the business *and* drive them to completion with engineering teams.

02 **Program Manager (“PgM” or “TPM”)**

Focused on process and execution. Accept specs handed off by PMs / PMMs. TPMs working with engineering and PgM’s with non-engineering.

03 **Project Managers (“PjM”)**

Similar to “program manager”.

04 **Product Marketing (“PMM”)**

How to present the product to the customer after launch. At some companies, they create a roadmap and hand it to PgMs.

04 **Partner Manager**

Negotiate and collaborate and develop roadmaps collaboratively with other companies, which often has significant communication and legal overhead and high-stakes “deals” negotiations.

These popular labels are all right-ish for “PM”

Captain of the ship

Voice of “the business”

Mini CEO

Voice of the user

The Extrovert

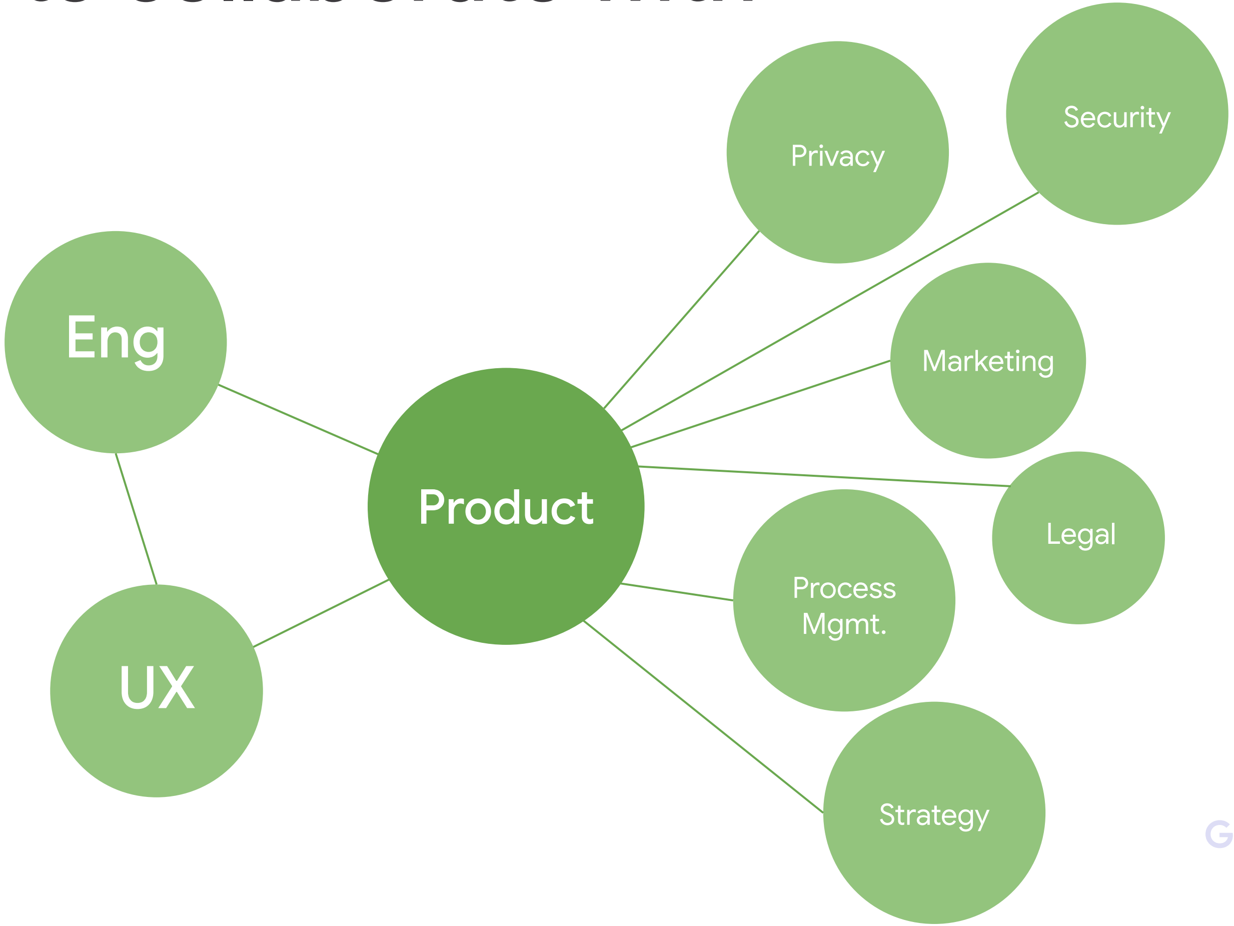
Conductor of the “orchestra”

Meeting Monkey

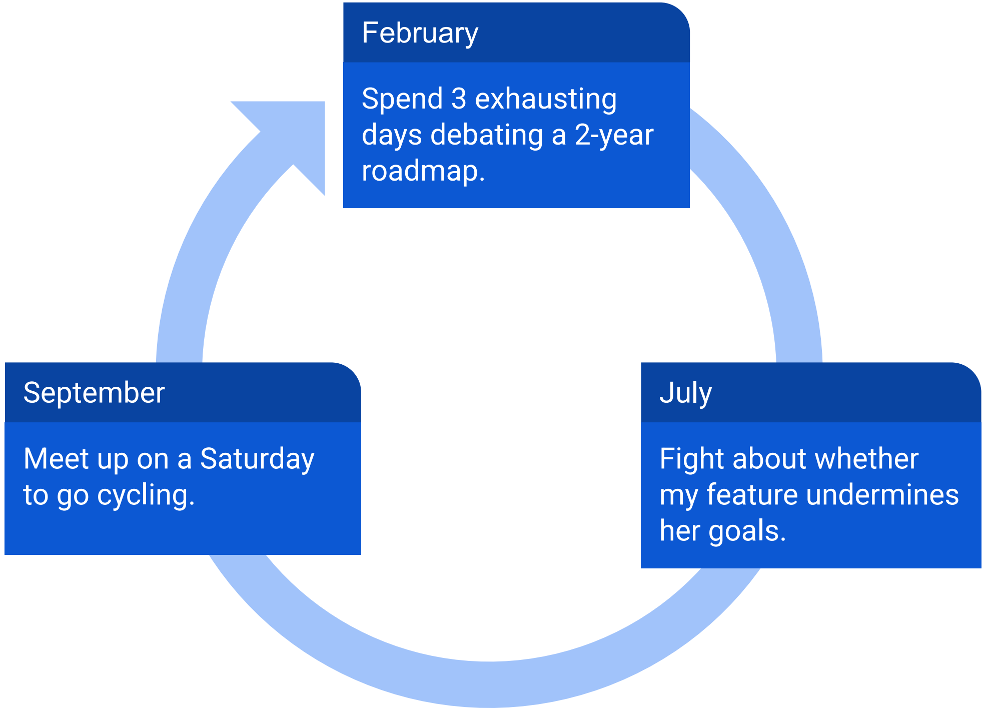
Face of the product

“Fire”-fighter

The PM role is to collaborate with *all* functions



Collab'ing well is important because relationship last *years*



February

Spend 3 exhausting days debating a 2-year roadmap.

July

Fight about whether my feature undermines her goals.

September

Meet up on a Saturday to go cycling.

And also important because you'll make no decisions without good relationships

You want it to stick the first time, not the third …

Make proposals that're readily acceptable to relevant functions, thereby becoming decisions.

Your "stakeholders" are the people who care about the outcome.

Ratification turns proposals into decisions.

You can “quiz yourself” to see if you’re a good collaborator

Do you talk with a person often?

Do you understand that person’s purpose?

Do you know what tasks that person often does?

Do you know when that person does a bad job?

Can you predict that person’s opinion?

Can you perform that person’s tasks in a pinch?

Do you let that person influence you meaningfully?

If you’re doing all this the people you work with, you’re collaborating well.

Luckily, effective collaboration is a learnable skill: communication tips below

01

Share context and info early and often.

What’s obvious to you rarely is to specialists in other areas. Even when you’ve shared before, it doesn’t mean people remember it or told anyone else. Repeating the basics again and again to anyone who will listen will serve you well!

02

Be clear and concise.

The less time you spend talking, the more time you have to *listen* and verify you were understood correctly.

03

Intentionally converge and diverge in turn.

In divergent phases, you’re making the project bigger: new questions, new goals, new customers, new ideas. In convergent phases, you’re choosing what feedback to ignore, what features to delay, etc. Being in the wrong “mode” frustrates collaborators.

04

Be appreciative.

Listening is the best form of appreciation. Make it clear you value other inputs and remember them later. Repeat what you’ve been told and give credit! And be extra thankful at key milestones.

Here's an example of how these might all look in one meeting.

Share context and info early and often – and be clear and concise!

Today we're discussing adding a thumbnail to our image files. We're expecting the thumbnail will improve the latency from opening the gallery app to seeing all your images populated.

Divergent phases add new ideas.

I know folks have been working hard investigating this problem. Does anyone have any new concerns about whether this would work? We should get all the risks tracked.

Convergent phases close out discussions.

There was an email thread a couple days ago about the thumbnail resolution. I looked at that and had some separate discussions as well, and we're going to set it at 200 pixels wide. Thanks for all the opinions there.

Remind people that you listened to people and appreciate them.

Jane shared a particularly good insight that the actual size of the thumbnail on-screen by default is just under a half-inch and the display is only 400 ppi anyway.

If you can’t remember any *skills*,
then channel this *attitude*: empathy.
AKA “consider your audience”!

Collaboration isn't always good – conflict is a normal part of bringing diverse experts together. How do we work through it?

01

Frame it as you both against the problem

You are not synonymous with your ideas, and you don't succeed or fail along with your ideas. Position the *decision* you have to make as a third party, and you can work together to defeat the decision!

02

Appreciate what you *do* agree on

Make the debate smaller and lower the stakes by identifying the parts you *do* agree on and setting them aside. It will also help you see you're actually on the same side!

03

Use friendly body language

You can disagree with respect. Show that respect by smiling, laughing, joking around, making eye contact, brightening your tone, and being energetic. Improving the attitude can improve the outcome.

04

Assume everyone has good intentions

People are rarely* *trying* to harm one another. If it *feels* like they are, it's *more* likely there's a benign cause. If you investigate what that cause could be, you might be able to resolve the decision.

* At Google, my experience is ~1 in 25 people.

End