# The Software Delivery Handbook



---

## Introduction

This Software Delivery Handbook focuses on how we, as a software team, organise ourselves to deliver software that meets our company objectives, without compromising on quality. It is, for now, largely agnostic on how we decide what to do. It assumes that the strategy is clear and **the inception phase is complete, so we know what we want to build and why**.

### Why do we need the Skyscanner Software Delivery Handbook?

As Skyscanner has grown, we've tried many approaches to working together to build software: we've experimented with tribes and squads, objectives and key results (OKRs), engineering principles, roadmaps, engineering health and, most recently, the Goals > Objectives > Strategies > Tactics (GOST) model. We've learned what works and what doesn't, and identified several bottlenecks to rapid, high-quality value delivery. **However, our approach is not always clear and well-documented.**

This can make it hard for new employees to get to grips with how we work, and it can hamper existing staff, particularly if they work across different goals and tribes. Writing it down means we can be clear on expectations. It also gives us a logical framework from which we can try out new ideas and make improvements.

This handbook is for those who make up the *Software Organisation* part of Skyscanner (the Software Org); the squads and tribes and everyone who has some role to play in delivering value through our software, data and supporting infrastructure (systems). It's also for anyone interested in how we go about delivering the product - our software.

### A single objective: deliver value that's high quality, fast

Our objective is simple: *for the Software Organisation to get better and faster at delivering.*

Specifically, at delivering desirable, feasible and viable product experiences and business capabilities, which match traveller and partner expectations and our business needs.

### A single KPI: reduce epic cycle time

Scope, quality and time; in software delivery, you can have any two of these, but not all three. As a business, it's rare that we are date-driven, so we focus on delivering the right features at high quality. However, without compromising on features or quality, there are always ways to optimise our machine and move faster.

Therefore we measure and optimise our **epic cycle time.**

An epic is a full unit of traveller or partner value. It's the smallest business value unit we use at Skyscanner. Each epic is made up of a series of user stories that contribute to the same full unit of traveller value (as opposed to smaller pieces of work such as feature enhancements). At the end of 2022, our typical epic cycle time was 80-90 days (see dashboard). That's almost three months (~12 weeks) to deliver one unit of value to travellers or partners. We should aim for this to be in the 3-5 week range most of the time.

Our KPI is to lower our epic cycle time, **without compromising on quality, or cheating** (e.g. creating tiny, pointless epics). In fact, focusing on high quality will help us lower epic cycle time by reducing waste and rework, but we'll get to that.


### A living document

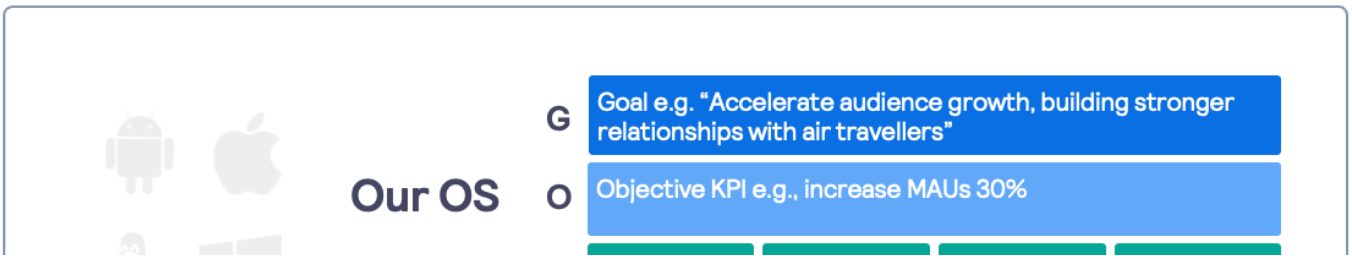
This handbook attempts to capture and document best practices at Skyscanner - but it **can and must evolve.** Anything we assert here can and should be challenged and iterated upon, with the ultimate objective of reducing epic cycle time.

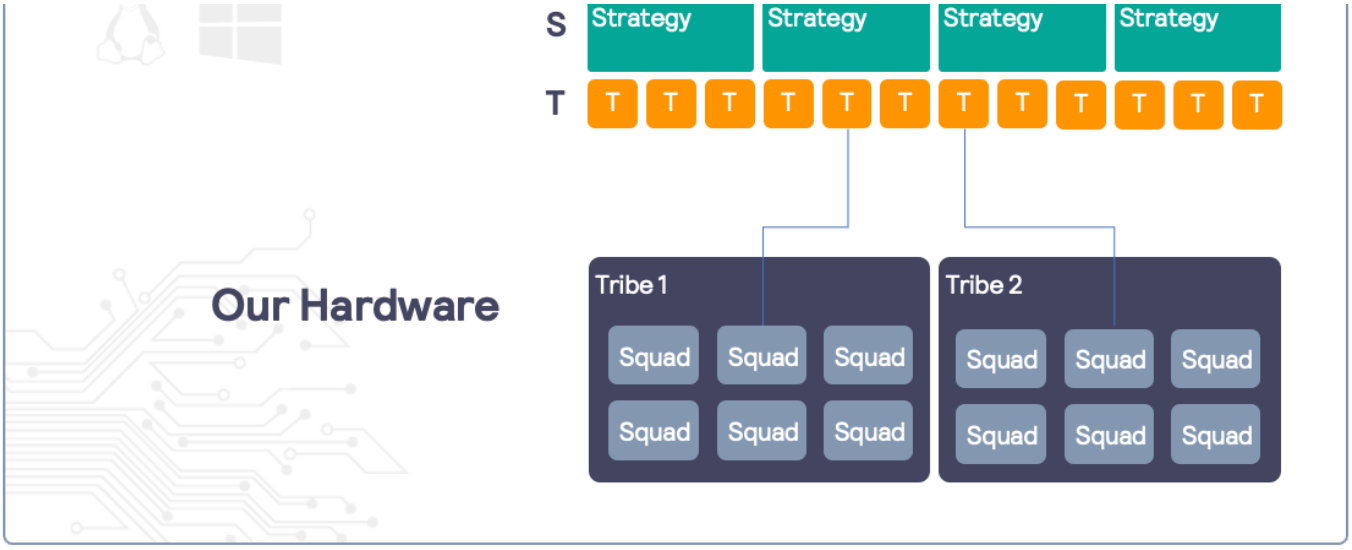
**The document is not a free-for-all Wiki**, but as you use it please feel free to challenge any assumption or directive - our software leadership team commits to evolving this document and keeping it updated.

As of 24 Jan 2023, our process for giving feedback to this doc is via the Slack Channel #handbook-q-and-a

## What to expect from each section

**How we organise ourselves** – This section explains why we are organised the way we are, and how the different pieces fit together and connect to our company strategy. It explains GOST, tribe and squad make-up, and strategy groups.






**How we plan & deliver** – Explains how we turn our strategy into the work the people in the Software Org do every day (and, therefore, how that connects back to our strategy). It explains things like roadmaps; breaking down work; the difference between proof of concept and minimal viable products; experience and system design reviews; quarterly dependency planning; how we maintain and prioritise product health; and how we share updates across the Software Org.


**How we build and maintain software** – This section covers how we make things and keep those things running effectively and efficiently. It includes our software delivery principles; how we build and deploy; how we maintain our software and data; how we maintain quality at the right level; and how we decommission it when we no longer need it.

**How we run squads and tribes** – This section explains the roles, responsibilities, events and artefacts we use to run our squads and tribes. It includes explanations of Tribe Vital Signs; the way we keep track of the improvements we need to make to tribe health; how the Software Delivery Ops team supports our delivery teams; the crucial partnership between engineering managers and product managers; and green flag.

There is a glossary at the end, but definitions will also be provided throughout the document where relevant.

---

## How do we organise ourselves?

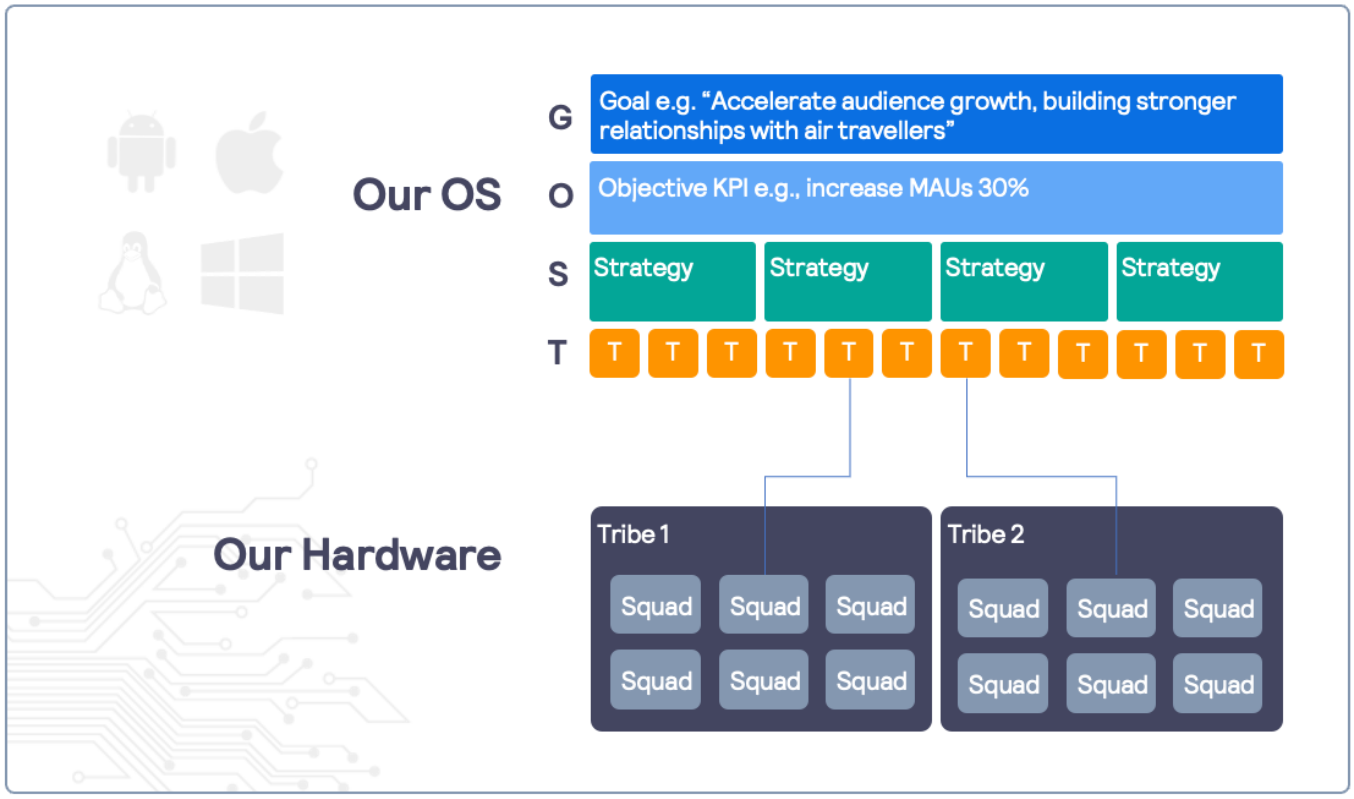
### A functional, journey-based system

Great companies come in all shapes and sizes, and all configurations come with trade-offs. At Skyscanner, we are organised by **function** (engine ers report to engineers all the way to the CEO, the same goes for product, marketing and so on) and also by **goal** (Audience, Trusted Marketplace, Sustainable Revenue, Understand, Enable and Empower), **but** we work together on **one cohesive product**. We do this because we believe our brand is best served by an integrated product experience that, for the traveller, is seamless. Organising this way optimises code reuse, and requires a high degree of collaboration. But when done right, the traveller sees one consistent experience, not the many teams behind it.

Note: This is quite different from many companies, typically segregated into business units which each owns a clear but separate product vertical (eg. flights, hotels, cars) and a profit-and-loss (PnL), meaning they are separate from an accounting perspective too. This makes accounting easy but produces a segregated product with high levels of internal duplication.

**Our "operating system" and "hardware"**

In any company, there is a connection point between the top-down business objectives and the code that is written and maintained. At Skyscanner, we use the **GOST** model to capture and manage our business objects, and **tribes and squads** to manage the code. Think of these like the company's operating system (the GOST) and hardware (the tribes and squads).




**The GOST - aka "the operating system"**

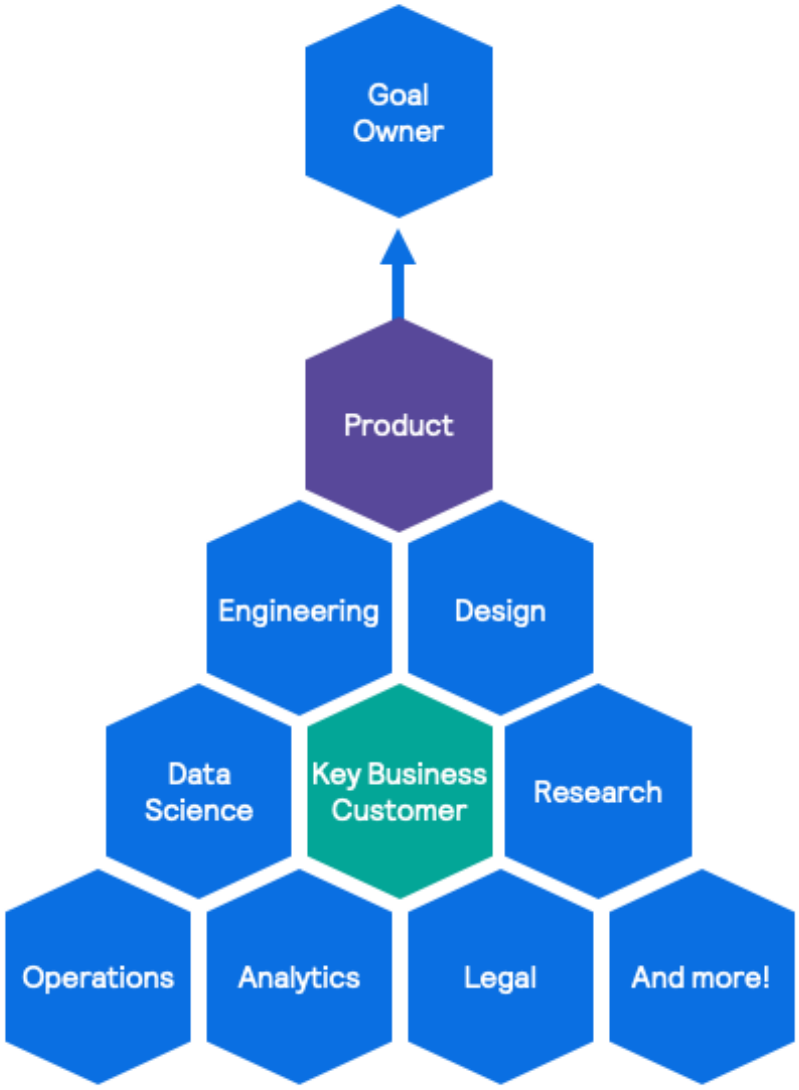
GOST stands for **G**oals, **O**bjectives, **S**trategies and **T**actics. It's a framework we use to organise our priorities.


Our GOST contains **six goals** that we believe to be evergreen, and very unlikely to change (even though the strategies within them might change yearly, and the tactics quarterly). There are three product goals (**Audience**, **Trusted Marketplace**, and **Sustainable Revenue**) and three enablement goals (**Understand**, **Enable**, and **Empower**). Here they are in a bit more detail:

1. Audience Growth & Retention - *Accelerate **audience** growth, building stronger relationships with air travellers*
2. Marketplace (flights & beyond flights) - *Build the most **trusted travel marketplace** for air travellers & partners*
3. Sustainable Revenue - *Create a **sustainable revenue** model*
4. Understand - *Understand our travellers, partners and business*
5. Enable - *Improve efficiency, effectiveness & resilience within the organisation*
6. Empower - *Build a leading tech company where people passionate about travel grow their careers and deliver their best work*

Within each goal are several **strategies.** For example, within Audience, we have multiple strategies, including "Affiliates Growth" and "Retention". Each strategy has a single **strategy owner**. This is very important. The owners can be from any discipline, but if the strategy requires software development, then the owner is a **product manager**.

All strategies serve a customer. Most of the time, this is "the traveller," but when the customer is internal or when the customers are our partners, we designate someone to act as the **key business customer**. This person will represent the customer and their needs. The strategy owner picks the strategy, but ultimately the customer needs to be satisfied with the outcome. It looks like this:




The strategy owner and key business customer then build a multi-disciplinary **strategy group** with all the relevant disciplines, for example, someone to represent design, engineering, commercial, finance, legal etc. This strategy group works together to lead the overall strategic direction for this area.

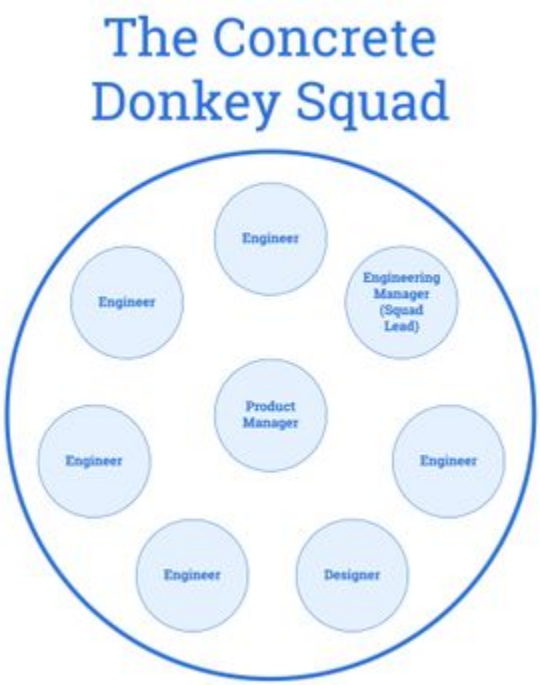
The members of the strategy groups are published in Confluence.

## The tribes & squads - aka "the hardware"

*NOTE: For this handbook, tribes are our **software delivery tribes.** Skyscanner uses the term tribes in other areas of the business too.*

**Squads** are the basic units of the organisation. Each owns a set of common **domains** (e.g. all the code and services associated with "SEO") and consist of:

- **6-8 engineers, including the engineering manager -** While product managers (PMs) own prioritisation, the engineering manager owns delivery and, crucially, must understand ahead of time when the team may block or be blocked by others and resolve these issues as best they can.
- **One product manager -** Most squads require a PM, and individual PMs can work across 1-2 squads in a similar domain. A few squads do not require a dedicated PM, but an engineer will take on the Scrum product owner role in those cases. PMs are responsible for prioritisation of the squad's backlog, ensuring the effective use of their capacity. The PM is also accountable for the health and quality of the whole product – not just the feature development. This includes managing technical debt (and *experience* debt - acknowledged shortcomings in the user experience), bugs, metrics, outages, customer satisfaction, etc., internally and externally. See here for an overview of the roles and jobs to be done within a squad.
- **One product designer -** In most cases, squads will have a single product designer who is a point of contact with the product design team. A design ops manager and a design team lead from the product design team supports them. The design team leader is accountable for the experience design over several squads; they support the designers in their area, coordinating the work and ensuring it aligns to create a cohesive experience. Other product designers may also contribute to the experience designs for the part of the product the squad looks after.
- Squads have **whimsical names** as they are not solely defined by their domain (e.g. "Pandas") - these can be tricky to remember, so Flightdeck is your friend!
- See our Engineering Organisation Design Principles for structuring, creating and merging squads.




**Tribes** are groups of squads created for organisation health and delivery optimisation.

- Tribes operate across closely related domains with higher dependencies on each other, e.g. "SEO" and "paid marketing". They are led by a single **tribe engineer lead**, **a tribe product lead**, and supported at least by principal product managers, principal engineers and senior engineering managers, one of whom is usually the **tribe health lead.** Tribes exist to partition software delivery work, coordinate across squads so they can deliver maximum value, and ensure squad health. In some tribes, they use Tribe Vital Signs to manage their health and drive improvement.
- Tribes are not directly responsible for strategy definition or delivery - that comes from the GOST, and strategies can sometimes cut across multiple tribes.


## How we plan & deliver

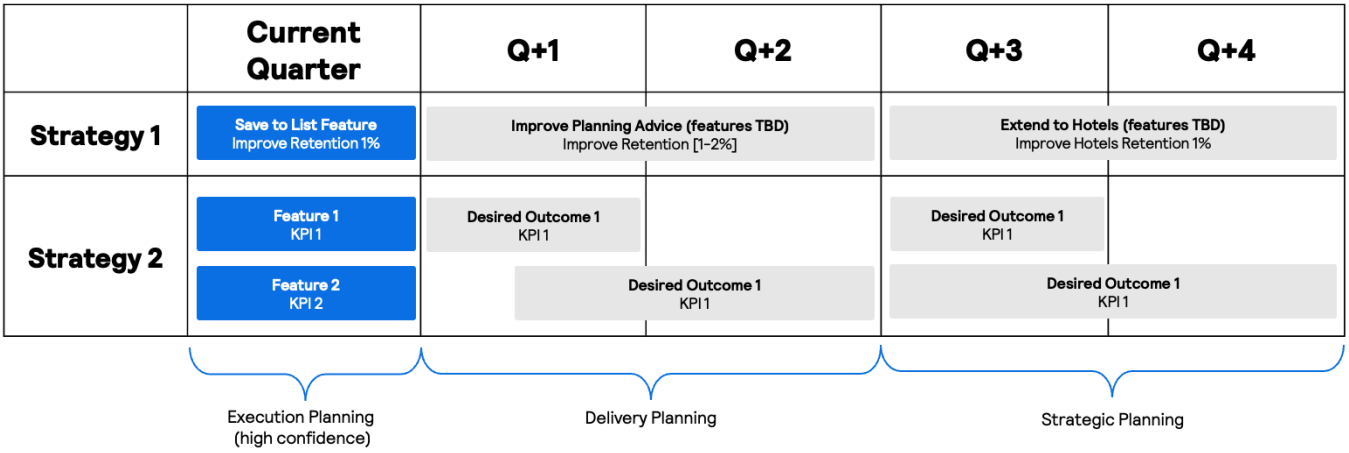
### Strategic planning & roadmaps

Each strategy in the GOST has a strategy lead constantly thinking about how to best deliver on their KPIs. The strategy lead creates and maintains a rolling roadmap of plans for their area; these are data-driven and traveller/partner-centric, informed by our innovation process. Our innovation process is a discovery method for uncovering unmet traveller needs and transforming them into actionable strategy briefs. It helps us collaborate across disciplines to add value for travellers, partners, and Skyscanner. These roadmaps are useful for planning, communications, and dependency mapping.

**Important: we do not operate in a sequential or waterfall way!** We are agile from our deployments to our highest-level roadmaps, which adjust every quarter. We do our best to plan accurately, but new insights and new priorities mean we are always prepared for change. Our roadmaps represent our best guess about the order of work and when the pieces will come together, but they are not explicit commitments to dates or what will happen several months from now.

**Plans for the current quarter should be high confidence.** When we enter a quarter, we need to plan accurately, as multiple teams, tribes and GOST areas will often need to work together to deliver value. So quarterly plans must be crisp, and we target a Say:Do ratio above 70%.

**The quarters beyond that, usually four, are indicative.** We plan for low work-in-progress (WIP) within and across squads and tribes, to ensure we can focus and swarm together to solve complex problems and ship the next most important thing. As a best practice, the further something is plotted to the right of the roadmap, the more it should be expressed as **outcomes** (“improve planning advice to increase retention by 1-2%”) rather than **features** (“ship planning tool 2.0”). Only when we get sufficiently close to implementation do we start converting roadmap outcomes into more tangible deliverables, again based on the jobs to be done (JTBD) identified through the innovation process.



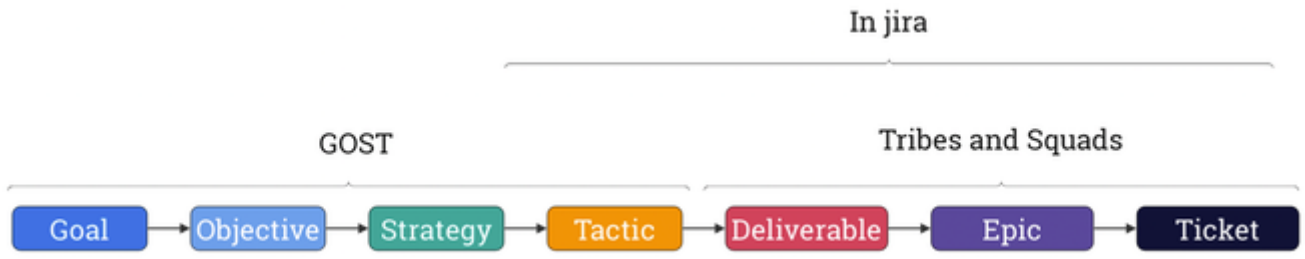

There are other sources for strategic work beyond the GOST structure, the most notable being the **Technology Adoption and Migration Plan.** This plan helps us prioritise the production standards we need to invest in upgrading, so we can meet our strategic goals and maintain or accelerate our pace of delivery. This work is important in reducing risk and epic cycle time. To help streamline prioritisation, we will publish the elements from key strategies that should be given high priority in the next quarter or more, at a company-wide level.

**Breaking down the work**

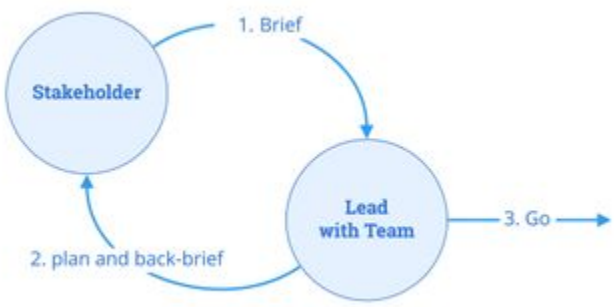
Each quarter, the strategy lead (typically a PM) works with the team and other GOST strategy leads to order and prioritise the tactics for the quarter.

**Important: we use the same system to prioritise traveller/partner features as we do product health work.** PMs are equally accountable for both, and work with engineering counterparts to balance moving forward on features with ensuring that we pay off technical debt or deliver on important technical transformations coming from our enablement goals (Enable, Understand, Empower).

The following diagram illustrates how we go from roadmap to epics.




- **Strategy**: Guiding principles, areas of focus or general statements outlining how the goal/objective will be achieved
- **Tactic**: A specific, defined programme of work with clear start and finish conditions (the dates may not be precise but the completion criteria should be clear on some level) which make a material contribution to achieving the objective and/or require a significant investment of available capacity
  - Read more on Definitions of GOST and 2025 Strategy Terms
  - We also currently have 'sub-tactics'; we intend to review the usefulness of this layer in the future
- **Deliverable**: A notable deliverable or milestone that features on the delivery plan for a tactic and contains epics. A deliverable is usually assigned to a single squad or in some cases several squads in the same tribe. As a rule, it does not generally exceed 12 weeks.
- **Epic**: A full unit of traveller/partner/internal Skyscanner value worth celebration and validation. An epic contains all the tasks needed to deliver a full unit of traveller value from the system, from experience design work to remediation work post-launch.
- **Ticket (Task/Bug/Story)**: A ticket may directly deliver user value on completion but often will not, as it will require other work to be completed to bring value to the end user. If so, it is deployed behind a feature flag, experiment etc. A task usually involves 2-3 days of work for 1-3 people who might collaborate by pairing or mobbing. Specifically, it might be a user story, a bug fix or a simple task.




We start by assuming the strategy lead knows what we should do next and has back briefed their engineering counterparts and everyone in the strategy group. See the Leading At Scale Refresher for more on briefing and back briefing.

For each tactic, **we need to feel confident it is the right thing.** Somet imes this is trivial, but sometimes we are not sure if our hypothesis is true or if the technical solution will scale, etc. In those cases and others, we sometimes build a **proof of concept**.

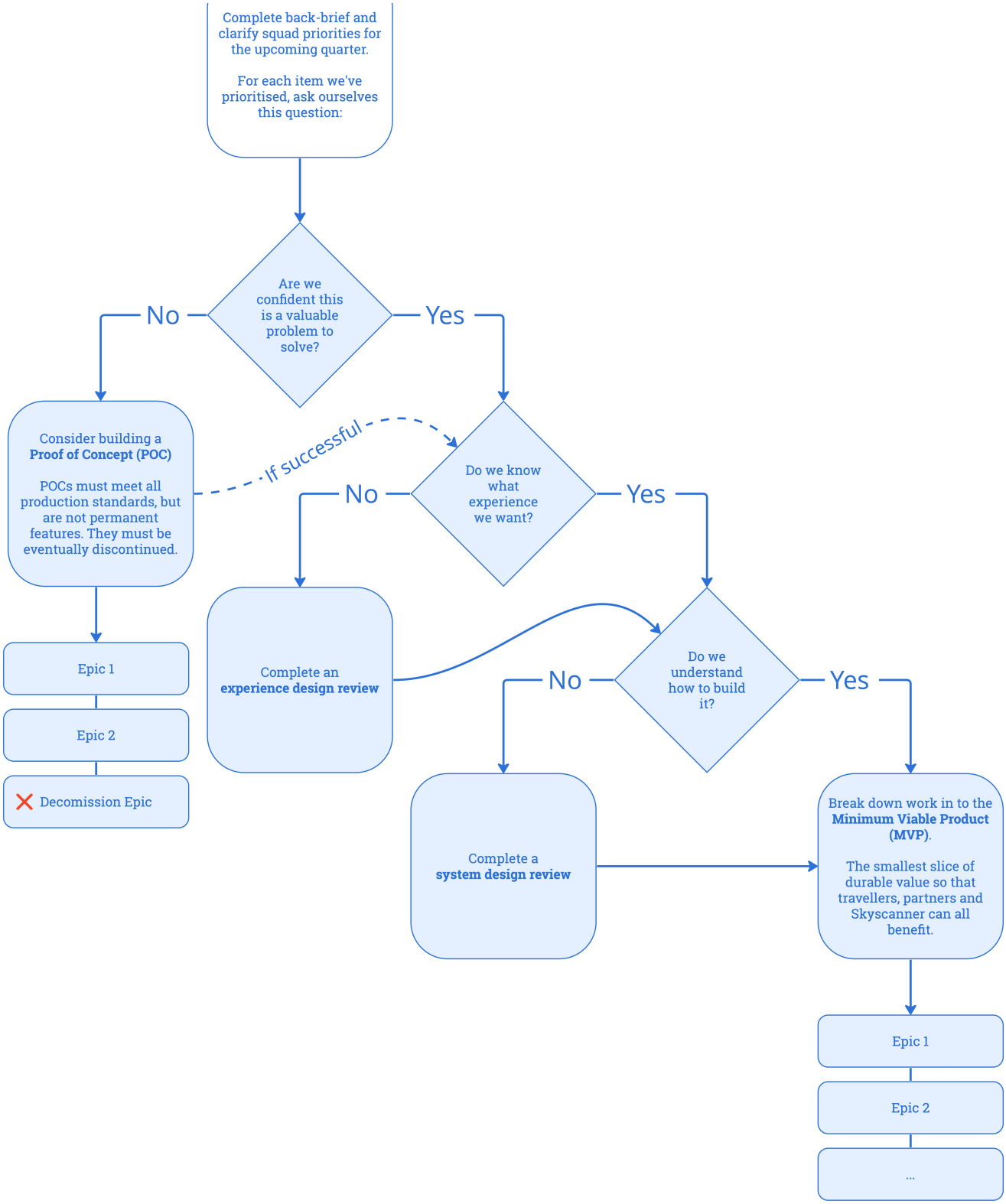
**Proof of concept (POC):** A POC is what it says on the tin; we're trying to prove or disprove that the concept we're building will work. Speed of learning is of the essence, and we accept up front that what we're building will be thrown away. We're building to learn, not to ship long-lasting code. This does not mean it's ok to ignore production standards! These are there to ensure our service is always stable and scalable. But it does mean it's OK to test the idea quickly, not use Backpack, rent data, hard-code resources etc. **We accept that even if the POC is successful, it will be discarded and replaced with a proper solution.**

Assuming we are confident we know what to build, we build in slices of a **minimal viable product**.

> **Minimal viable product (MVP):** This is the smallest slice of durable value we can release so that travellers, partners and Skyscanner can all benefit. The key with an MVP is to figure out the smallest thing we can deliver that will generate value and longevity. If the MVP contains user experience changes, we require them to be brought into Backpack first, just as the rest of the MVP must also meet all production standards from the outset. An MVP should be able to **survive on its own once shipped**, even if we do not get around to adding more value to it.

It helps to be clear which of these two modes we are using, to drive this we will label the highest level ticket (deliverable, epic etc) with either MVP or POC so it is clear to everyone involved.

## Simplified Planning Process




Once we have defined the next MVP slice to build, we need to understand ***how*** we will build that slice (and possibly how we intend to build future slices). There are two parts to this:

**1. Do we know what experience we want?** Again, sometimes this is trivial - the change might be backend only, or the UI changes minimal and can be expressed using Backpack. But sometimes, we are exploring completely new areas, so we might need to employ the full product development flow to explore options. Whatever the case, if major UI changes are needed, we run an **experience design review**.

**Experience design reviews**Product Design Reviewsare used whenever we design something for humans to use (so yes for a web interface but not for an API). An experience design review defines the general approach for UI and UX, setting the standard for the https://skyscanner.atlassian.net/wiki/spaces/a11y of the target experience. They typically involve designers, user researchers, front and back-end product managers, and front-end developers.

2. **Do we understand how to build it?** If the work is not trivial, we set up a system design review.

   **System design reviews** are used whenever we are making non-trivial changes to our systems. Generally**, any work that comprises a whole epic or more will require and benefit from this process**. As with experience design reviews, they are a multidisciplinary process, although mainly carried out by engineers and product managers. They are where we share the approach we will take to building, deploying, and validating the system that will meet the desired outcomes. They are our primary opportunity to capture all the key non-functional requirements (NFR) - or qualities - like security, accessibility, data management, running costs, scalability, end-user performance and so on, as well as the core functional/experience/business requirements. Any exceptions to production standards must be identified at this stage, with the relevant senior principal engineer confirming any exceptions.

Both processes must include representatives from all the squads and tribes that are expected to contribute to the implementation, so they can input the approach and the subsequent planning. The Product Development Flow also lists other groups across Skyscanner that may need to be involved in either of these reviews.

This means that when we start building, we do it with as clear a design as possible for both the internals of the system and the user-facing parts of the experience. Still, there will be unknowns (both those we know about and those we do not) so the design work must be continuous, and we must prioritise the risky and complicated aspects first and be prepared to change and adapt, keeping those involved in the original design work (experience and system) involved with any changes.

The resulting work is captured in one or more thinly sliced epics, each expected to be delivered in under (roughly) five weeks.

We define a set of related epics as a **minimal viable product**.

Once the work is agreed upon, we add fine-grained tickets to the epics, which get prioritised on a rolling cadence for the squads weekly.

This level of planning helps identify dependencies. Whilst some work will require squads to deliver sequentially, we strive to plan and design to allow squads to deliver in parallel, in vertical slices, to surface integration issues early.

First and foremost, this process is collaborative, with the whole squad involved at some point (the earlier, the better). It's about challenging ourselves to imagine what can be done in this short time to deliver value somewhere in the business. Our BizOps team may be able to provide support for coordinating this planning activity, especially where several tribes are involved. As new insight is acquired from customers, stakeholders and squads, it's inevitable that plans will change. As such, it's important not to spend too much time planning too far into the future. But working out that sweet spot for any given plan is more art than science.

### Aligning the dependencies

Ideally, all the work can be delivered by the relevant tribe. However, often a tribe will have dependencies on another to complete an MVP.

The outcomes of all the GOST-level planning then produce a set of asks for each tribe/squad. Here the tribe leads, product and engineering managers work to untangle dependencies for the coming quarter. This might mean aligning backlogs carefully, agreeing on dates for deliverables, or making hard decisions about what to deliver and what to push back. This might result in a prioritised tribe-level backlog which squads can then align to. This is useful when the squads in a tribe generally work closely together and so need to be working on the same things.

Most of the time teams can agree, but sometimes they need help prioritising. For this, we have a quarterly dependency meeting to align dependencies and priorities across the company.

To make good decisions, the purpose of the work (connection to strategy) and probable costs (time, other work not done) must be clear. The asks of other teams must be crisp and costed, not just "I need four weeks of your time," and need to be prioritised with any other asks being made by the same GOST area.

The dependency meeting is where **the train leaves the station!** Once the decisions are made, we try hard to not change things during the current quarter without good reason and consideration, as this can be very disruptive.

### Execution coordination

Our product managers lead squads, designers and others in progressively adding detail and definition to the backlog at the ticket/user story level for each epic they have outlined. As squads create tickets for the epics in the backlog, they back brief their plans and understanding of the asks to their PM, and likely to other stakeholders and colleagues. This approach helps each squad validate whether they will build the right thing to the right quality, and shares any risks they've identified. PMs prioritise the backlog to front-load validated learning, risk mitigation and key areas of user value, while maintaining sufficient levels of product health (Future of Eng Health Time > Product Health and Maintenance) to enable a sustainable pace in the long term. Tribes and some cross-cutting programmes will operate 'scrum of scrums' forums, where leaders from squads and tribes come together to align and coordinate day-to-day/week-to-week activities with each other.

Product managers also create and update a highly visible plan that is shared with everyone involved. It includes progress updates on the status of all the work needed to achieve the outcomes of the epic; designing, coding, testing, running experiments and so on. Where cross-organisation coordination is required to launch a product or feature (for example with other software teams or beyond) this is particularly important.

**Reporting on progress**

We report progress/blockers via the GOST in the monthly/quarterly business reviews (M/QBR); these usually involve an internal GOST leadership review ahead of the wider exec-led M/QBR. We celebrate shipping things and removing things that have a measurable impact on travellers, partners and Skyscanner - see the #what-we-shipped and #what-we-shed channels for updates. We can also choose to share progress updates with the wider team by sending emails to GoalUpdateSummaries@skyscanner.net.

---

## How we build & maintain software

At Skyscanner, we aim to build a world-class software delivery team. To do this, we follow six simple software delivery principles. These provide a framework that we all can work from that ensures we start small on any change, value quick iteration and fast feedback, and, ultimately, deliver high-quality change that is sustainable (to maintain) long-term.

In this iteration of the principles, they move from being "engineering principles" to "software delivery principles," the difference being that these are for everyone involved in software delivery. As such, anyone involved can give feedback that helps iterate them.


### Skyscanner's Software Delivery Principles

**We have a clear definition of success for every piece of work**

- Any work on the backlog is assigned an anticipated value and expected cost of ownership – these are measured by specific quantitative metrics, so it's easy to see if and when they are reached
- We stay focused on outcomes throughout delivery. We attach objectives to tactics, deliverables (whether a POC or an MVP) and epics; these items are not complete until the outcomes are achieved or abandoned. For example:
  - A price accuracy project might meet the outcome of its MVP when 95% of prices shown on day view are within 1% of the final price paid
  - A flight search performance improvement project might meet the outcomes of its MVP when 95% of the first results are returned in under one second; it reaches its second milestone when 95% of the first results are returned in under 500ms.

**We follow our data policy to preserve traveller, partner and Skyscanner trust**

- The outcome we want is trusted quality data providing a single version of the truth that can be used as a lever for Skyscanner's growth

- Appropriate data security, consent, tracking, logging, storage, processing, reporting, etc. is fundamental to reducing or mitigating threats, and protecting Skyscanner's business and reputation
- Data policy covers many aspects and the following areas are actively being worked on at Skyscanner:
  - Data Production Policy: This will contain standards for producing data and where to produce it to so that it can be transformed into usable datasets. It will include how to apply classification so we know which data needs to be treated in a 'special' way
  - Data Contract - Template : A formal agreement between producers of enriched/aggregated data and their consumers
  - Data Quality/Health: Expectations for data quality and how we manage and maintain it to ensure it is trusted
  - Data Consumption Policy: How to consume data, where to consume it from
  - Data Privacy Policy: This includes our data deletion policy

Our data contract with consumers and data privacy policy are already in place, others are expected to be in place by the end of Q1 2023.

**We deliver using our defined production standards**

- Production standards exist for every area of technology we operate in; our production standards are not optional and include using our design system, Backpack
- The appropriate executive stakeholders must approve exceptions to standards
- Anyone can propose changes to our standards, but they must be approved by executive stakeholders as the knock-on impact and implied work from a production standard change can be widespread

**We drive velocity by maintaining high quality**

- Quality varies according to the classification of the work, for example, whether it's a proof of concept, MVP, or long-term/foundational work
- Engineering quality (unit tests, etc.) drives velocity by creating fewer distractions, delays and interruptions and minimising rework
- Product quality also drives velocity by ensuring we build the right thing and that it delivers consistent value to the customer

**We remove complexity whenever possible**

- Sharing contracts, code libraries, APIs and our design system remove complexity by reducing duplication and providing one way to do something
- Creating checklists to standardise processes makes it easier to build and operate systems the same way every time
- The similarity of design and implementation – following patterns and conventions, etc. – removes complexity by making disparate systems familiar to everyone; metrics charts, code formatting, alert names, and so on

**You build it; you run it**

- Squads are responsible for the operation of the services assigned to them; ensuring they are built to and maintained at our production standards, and according to these principles.

## Build and deployment

We're here to deliver value to our travellers and partners. By using off-the-shelf and pre-built technology, we avoid reinventing the wheel and allow our team to focus on building the tech with the biggest impact.

To do this:

- We iterate using feedback on our approaches through experience and system design reviews
- We use a single language for each domain (frontend, backend, data)
- We have strong API contracts and shared libraries (based on gRPC/Protobuff)
- We make use of reusable frameworks and components that everyone can benefit from
  - We use and evolve with our Choosing language and technology ; Java, NodeJS, Python and AWS Lambda
- We have clear and concise service, system and method naming, see this blog
- All services are registered in Tower to help with identification and re-use
- We build iteratively with small pieces of end-to-end functionality, learning as we go
- We build with accessibility, testability, data and security in mind, using the knowledge of our subject matter experts in each area
- We lead with an experimentation-driven approach, setting out the effect we expect our changes to have and testing them, before considering any change worth keeping and maintaining. This means PMs prioritise testing the riskiest/unknown parts of the build first; we raise any hard trade-offs we find between traveller/partner experience and Skyscanner's interests to our #hard-tradeoffs group for a quick confirmation; and we abandon builds when sensible
  - We recognise that, historically, we have not run enough experiments. Our current goal is to undertake ten times the number of experiments undertaken in 2022
- We deploy safely using feature flags and blue/green versions and make sure we have followed the guidance in the Product Development Flow as appropriate

**Quality**

Everyone that touches code or makes a coding change at Skyscanner is responsible for its quality. Every product change undergoes a peer review process, and quality is the most important consideration. How we achieve this is optional; some prefer pair programming, others mob, and in some cases, an async review will be more appropriate. No matter the method, the two (or more) humans that review it are jointly responsible for the quality of the resulting end-user experience and underlying system.

We aim to work in a way that engenders quality. This means the changes should:

- Be small enough to be reviewed in an hour (fewer than 100 lines of code). Anything bigger needs broken up and put behind a feature flag
- Be self-documenting, so you don't need a manual to understand the change and how it fits into the wider system
- Be tested with a minimum of unit, integration, functional and system/integration tests, but we prefer to start with TDD (test-driven development), with more advanced squads aspiring to use BDD (behaviour-driven development)
- Meet non-functional requirements in areas like security, accessibility, cost, observability, and performance/load
- Meets our production standards, see above

#### Maintenance and sustaining the product (operational excellence)

We pride ourselves in the quality of our products, but we can only build quality products on strong foundations. Improving our products is important, but we also care about maintaining what we've built and making sure it's well looked after.

To do this we:

- Hold ourselves accountable to the promises we've made to our users through agreed service level agreements (SLAs), objectives (SLOs) and indicators (SLIs)
- Take time to ensure our tech stack is upgraded responsibly; for example, we are only one major version behind language or software updates. We are considering how far behind it is responsible to be on our Backpack design system, as even one major version behind potentially introduces UX differences for travellers
- Ensure we monitor our product stack costs using CloudZero, addressing bugs through Jira, vulnerability using multiple security tools, and regressions using NewRelic, when they happen
- Address technical and experience debt
- Respond to incidents and bugs, and fix security vulnerabilities
- Identify ways to improve our epic, ticket and code (using Plandek)
- Share learning through our incident learning debrief (ILD) process, including cases of near-misses and success stories. Each tribe maintains their section of the Tribe ILD Library for future reference.

Even though we hold a high bar for quality, there is still the potential for incidents. To handle these effectively, we:

- Follow our 1. Incident Management Process , which everyone in the Software Org should familiarise themselves with
  - Follow the ILD process for all incidents, even if the incident seems 'minor' or there are many other priorities in your squad
  - Revert systems to their last known good state wherever possible rather than fix-forward. This includes rollbacks, disabling experiments or reverting feature flags - and we should always build and deploy software to make this as easy as possible. If we have to resort to a fix-forward, the ILD should explicitly address how we will avoid that outcome in the future
  - Continually improve our alerts and monitoring capabilities, aiming for no false positives and no acknowledgement without action
  - Include prioritising ILD actions as part of product managers' accountability for the quality of the whole product, even where that means reprioritisation of other important work, including feature development

Services are categorised into whether they are supported 24/7, with an out-of-hours (OOH) rota for incident response, or only supported during business hours by the green flag process. OOH rotas may be scoped to a single squad or shared between multiple squads for services which are critical or have multiple squads contributing to them. To ensure the best experience for on-call engineers:

- We have guidance written for the roles and responsibilities of OOH work at Working Out of Hours @ Skyscanner (aka 24/7 aka On-call); engineers on OOH rotas and squad and tribe leadership should familiarise themselves with this
- To load balance appropriately, we should aim for a minimum rota size of 6 (ideally larger). All engineers working on services with 24/7 support should be part of an OOH rota by default. However, we acknowledge there will be people for whom this is not possible (either permanently or temporarily) and will accommodate this
- We treat any alarm outside of business hours as requiring an ILD. Waking people up at 3 am should not be normalised! If it was a false positive, we must review our alarms

#### Decommissioning software

When it is time for something to be decommissioned, we do this responsibly, cleaning up as we go. We use the Service Decommission Checklist t guide this activity and then celebrate the simplification of our product on #what-we-shed - and sometimes also with cake!

---

## How we run squads and tribes

### What exactly is a tribe?

The main function of a tribe is to support the delivery of the GOST and guide our squads. To do this, they use:

- Tribe Vital Signs to monitor and improve the health of both squads and tribes. This is usually the role of the tribe health lead
- Scrum of scrums, to support and coordinate week-to-week delivery (when squads are working together)
- Ongoing reviews of individual performance to support personal development
- Operational Excellence Reports and risk management to maintain a low toil and risk profile
- Incident learning debriefs as needed, to share learnings from incidents across the tribe and beyond
- Frequent system design review sessions to ensure awareness and alignment of key planned changes
- A hiring plan to make the best use of the tribe's assigned headcount
- Tribes also coordinate their squads' approach to quarterly planning, making sure escalations are made about dependencies and blockers

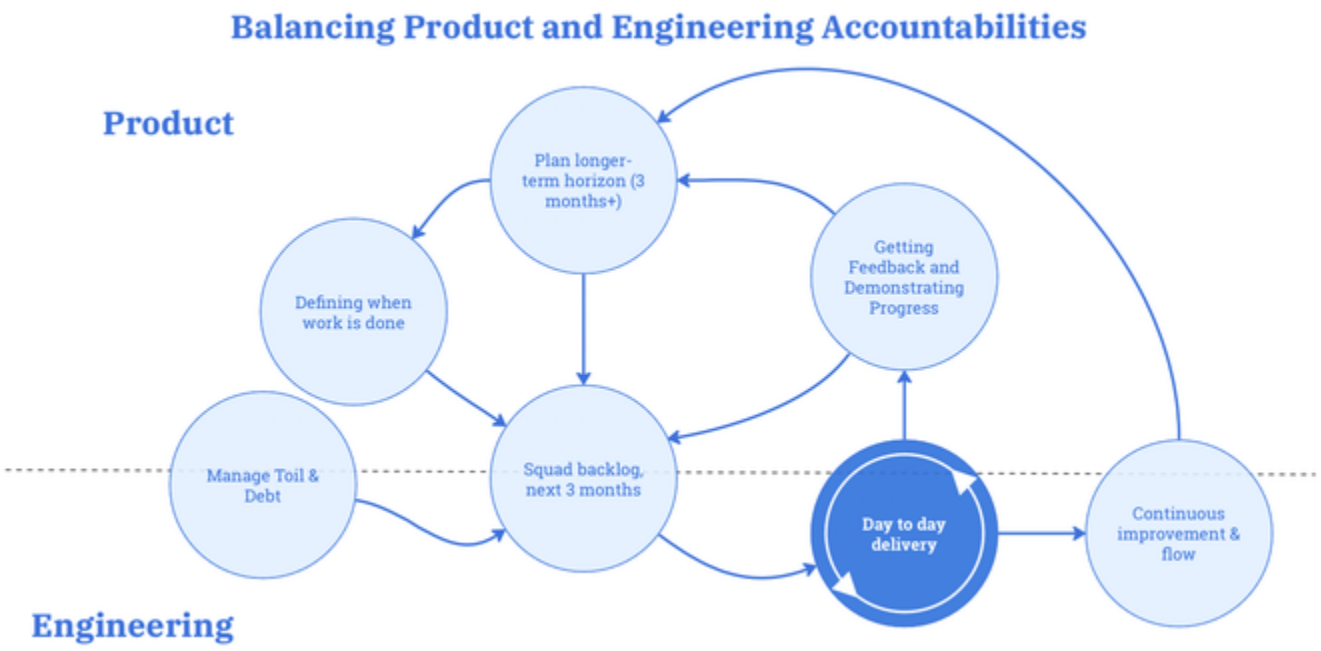
#### Further support for squads & tribes

- The practices in this handbook support sustainable and effective delivery at Skyscanner. Everyone involved in software delivery should support this way of working and contribute to improving it.
- Our Software Delivery Ops team can provide additional support if you need it. They work to create productive environments within our squads and tribes based on mutual trust and growth. This support is provided in different ways:
  - Reactive: Working with squads and tribes to improve and resolve specific issues through a mix of coaching, facilitation and mentoring
  - Proactive: Driving, guiding and supporting the organisation and the people in it through key strategic changes to our ways of working
  - Operational: Supporting certain processes that are needed to maintain a healthy organisation (e.g. engineer hiring, learning & development)
- We drive the growth and movement of our engineers by encouraging and enabling Engineering Internal Mobility - Guidance
- We support growth and provide learning opportunities for everyone to develop the skills needed to support and guide critical collaborations in meetings, workshops and other spaces, which also helps set up our future leaders.

#### Hiring & allocations

- Each GOST area has a portion of the total fixed engineering headcount, and this is agreed upon by the executive team as advised by the execs that lead the software team. There is no additional headcount and headcount increases are usually only granted on the basis of adding new capabilities.
- The engineering and product leaders in each GOST area are responsible for assigning this headcount to the tribes within that area.
- Tribe leads take the information in the Eng People Scorecard to gauge if they are planning the right number of roles and vacancies to maintain their tribe near their budget.

### How we run squads, role flexibility and jobs to be done




At Skyscanner, we pride ourselves on being flexible and adaptable. This section gives guidance on who might be accountable for different activities, for example, a product manager, but this can be flexible from one squad to another. In squads that do not have a product manager, accountability usually moves to the engineering manager or sometimes a principal engineer. In cases where all the roles are present, they must work closely, leading and supporting each other.

Those closest to the work have the best context to figure out how to do it effectively. Therefore, we don't prescribe exactly how a squad should be run, for example, by asking everyone to follow the Scrum Guide to the letter (although we do recognise it as a good default and offer a training

course on Scrum for anyone who wants to learn more). It's more important that anyone making decisions about running a squad understands *why* things are done the way they are. It makes sense for us to set out the basic qualities or capabilities a squad needs, and share advice on how these might be achieved and who the responsibility generally sits with. We can then give squads the flexibility to improve their capability in whatever way they see fit. Those qualities and capabilities are listed here, with further links to our guidance and training.

See here for a handy checklist of the roles and jobs to be done within a squad.

**Manage toil and debt**

Managing toil and technical debt is key to delivery velocity in any team. As the squad has one, collective capacity, everything they do must be prioritised at the same time. As the person responsible for the team's outcomes, the product manager trades off delivering something today versus how fast we might deliver something in the future. Occasionally, this will mean choosing to incur technical debt at the cost of future speed. This is ok, but that cost must be paid in full at some point and deviations from production standards need to be widely reviewed and validated. The product manager may not be best placed to understand the impact of historical decisions that have incurred debt, or opportunities that may be available to increase velocity in the future. Therefore, the engineering manager is responsible for assessing and sharing this information with the product manager. If this is done well, the squad should be able to spend most of their time focused on strategically impactful work, since the product is continually easy to extend and has low toil.

The engineering manager is also responsible for flagging mandatory product maintenance activities to the product manager so they are prioritised as needed, as is the team's designated designer for experience debt/issues. The whole team is responsible for seeking and identifying opportunities as well as risks that impact the squad's productivity and effectiveness.

Reference: Future of Eng Health Time > Product Health and Maintenance

**Defining when work is done**

Defining when work (or its parts) is done is key to limiting waste and rework, and enabling incremental progress and delivery. There are two elements to consider:

1) the *acceptance criteria* of a given piece of work and

2) the *definition of done* squads use on all pieces of work to keep the product quality consistently high

Product managers may commonly delegate writing acceptance criteria to the team; that said, they remain accountable for ensuring the scope is well defined, supporting engineers doing this work if that's the approach in the squad. The definition of done is something the team develops and continuously improves together. The engineering manager is responsible for curating it and consulting with the product manager to ensure it's aligned with their quality needs.

- Further guidance on knowing when work is "done"
- Further guidance on producing and maintaining high-quality products/services

**Planning for the long term (three months plus)**

The product manager leads and, as with toil and debt, they partner with the engineering manager and the team to create a long-term plan that includes engineering and product tasks. This gives the squad and their stakeholders a clear view of the roadmaps, their feasibility and the practicalities of delivering them. The engineering manager is responsible for highlighting likely dependencies between tribes and squads, so that the product manager can arrange support from other squads/GOST areas.

- Further guidance on planning towards the longer-term horizon

**Squad backlog (next three months)**

As with longer-term planning, the product manager is responsible for creating a prioritised and well-scoped three-month plan that clarifies what needs to be done and why. The engineering manager ensures this delivery plan is accessible in Jira (Plans) and considers the squad's capacity, taking into account learning and development, cultural activities, personal time and anything else that needs to fit alongside our delivery work. The engineering manager is also responsible for aligning with other engineering managers on dependencies between squads. This means proactively looking ahead to ensure smooth execution, and escalating roadmap conflicts to the product managers. In addition, the engineering manager makes sure our system design review process is followed.

- Further guidance on sprint planning and improving dependability

**Keeping work-in-progress low**

Product managers and engineering managers should keep the number of concurrent epics as low as possible so the squad maintains focus on completing them together and moving on. A rough range of between one and (n/2)-1 can be useful (where n is the number of active engineers in the team). This approach encourages more ensemble work (pairing/mobbing/teaming), which generally reduces interruptions, context switching and rework. It also improves team resilience and is good for personal development, as it allows for more frequent knowledge transfer. Fundamentally, we start with the assumption that epics in progress should equal one, and then challenge ourselves as to why it needs to be

higher at the expense of delivering the highest priority epic. For fast execution, the GOST area lead and tribe leadership should minimise the number of strategies and tactics that run at the same time. Making this plan visible helps raise awareness of the concurrency we are operating at, and gives anyone a chance to call it out.

#### Day-to-day delivery (this week and next)

The product manager makes sure the squad has two weeks' backlog ready. In practice, they may delegate the refinement of this backlog to engineers, but they need to be available for questions and clarification and to make prioritisation decisions. The engineering manager is responsible for the squad's day-to-day delivery: planning, refinement, estimation, delivery of iteration goals, quality and continuous improvement. They are responsible for the effectiveness of the team process and facilitate ceremonies (this can be delegated to other squad members too). Engineering managers also proactively manage dependencies, collaborating with any other squad they need support from.

- Further guidance on setting squad working practices

#### Managing interruptions and supporting other squads – green flag

We all need to support each other in getting work done by answering quick questions or reviewing simple PRs (pull requests). In squads, we usually designate a single person on any given day to be the person with the 'green flag' for interruptions (this goes back a long time and started with physical flags!) This role, which should be rotated at least weekly, is key to letting everyone else focus on the squad's work. That said, the green flag should also be able to exercise discretion when picking up asks, considering how urgent or important the requests are and whether they might be better planned properly and prioritised by the product manager in the next sprint planning session.

#### Writing and reviewing Jira tickets

Product managers are responsible for effectively defining backlog items - each should contain the information needed to understand its scope. Entering work items in Jira may be delegated to engineers in the squad. Regardless of who enters the information, tickets will be reviewed by the product manager and squad engineers in backlog refinement sessions. This is to make sure they contain all the information needed to be estimated and actioned by anyone in the team, and to lower the chance of it being blocked because more discovery or definition work is needed. Each squad defines what makes a ticket ready to be implemented.

#### Getting feedback

The product manager is responsible for getting customer feedback and sharing it with the team, for example by working with user research and user satisfaction. As user feedback can come in data, the product manager prioritises work to implement data logging and presentation where needed. The product manager also engages with the product daily to ensure the work-in-progress meets requirements and offers a great experience to users.

- Further guidance on seeking out and acting on feedback on work
- Further guidance on openly demonstrating progress each sprint

#### Continuous improvement and flow

Monitoring how the squad is working and attending to the human system that the individuals come together to create is everyone's job, however convening the conversation, ensuring there is space for it and that it has the right inputs will most commonly rest with the engineering manager. They will be looking at quantitative metrics like flow and DORA, employee engagement survey results and the more qualitative discussions that happen ad hoc, in retrospectives and around squad health checks. The product manager is often in a unique position to give an external perspective and add context to the squad's efforts to improve.

- Further guidance on continuously improving your ways of working
- Further guidance on monitoring and improving the flow of work

#### Note

The responsibilities outlined in this handbook are only part of the roles of engineering and product managers. Here we have focused on the interaction between these roles and the key responsibilities for the squad's operation. There are other parts to these roles at Skyscanner. For example, a product manager can be a strategy lead or line manager in one or more squads.

---

## Glossary and references

*(Ideally, this references a single Skyscanner Glossary, though that does not exist yet)*

- Acronyms should be expanded on first mention, then you can say “MVP”
- Each term should be **explained** the first time it is introduced in a bubble

- **Domain:** A logical grouping of components, responsibilities and plans owned by a single squad.* Within each domain, priority calls can be made according to easily comparable values.
- **Incident Learning Debrief** (ILD): A short write-up of the root cause (also known as the “5 whys”) of a production incident, intended to create shared learnings and to avoid others making the same mistakes
- **System Design Review** (SDR, aka DR): A living documentation of the current state of a service (or group of services). Created as part of the initial phase of a new, large piece of work and kept up to date with any changes.
- **Monthly Business Review** (MBR): Every month, the executive team look through each GOST area and review progress, blockers and plans for the following month.
- **Quarterly Business Review** (QBR): As MBR, but quarterly, more focus is given to the following full-quarter roadmap and dependencies
- **Squad Lead** (SL): An engineering manager who runs the day-to-day operation of a squad, albeit often delegating the “running” of ceremonies, etc. to help grow their teams.
- **Goals, Objective, Strategy, Tactics** (GOST): A commonly-used strategic framework for aligning the company behind our vision and mission. Replaced OKRs.
- **Objective and Key Results** (OKRs): Another commonly-used goal-setting framework, used to define measurable goals and track their outcomes. Used at Skyscanner until Q4 2021, popularised by the management author John Doerr
- **Backbrief -** leaders should invite their directs to *backbrief* their understanding of the requested goal and their plan to them and their peers; this builds confidence in the plan, the team and their understanding of the goal with everyone. Read more Leading At Scale Refresher
- **Say:Do Ratio:** The number of things a squad says it will do versus the number of things they deliver
- **Test-Driven Development** (TDD): An iterative approach to development which follows a test and build cycle
- **Behaviour-Driven Development** (BDD): Software development that addresses the needs of the business and the end-user
- **Service Level Agreement** (SLA): A commitment between a service provider and a customer
- **Service Level Objective** (SLO): Part of an SLA that addresses a specific metric, e.g. uptime
- **Service Level Indicator** (SLI): A metric used to measure SLOs
- **Product Manager** (PM)
- **Jobs To Be Done** (JTBD): A framework for viewing the products and solutions we provide in terms of how they help our users achieve jobs they want to do
- **KPIs** – Key Performance Indicators
- **PnL** – Profit and Loss
- **MAUs** - Monthly Active Users
- **BPU** - Bookings Per User
- **CVR** - Conversion Rate
- **ARPU** - Average Revenue Per Users
- **gRPC** – g(Google) Remote Procedure Call