mateco IT Product Lifecycle



00

Product vision

PRODUCT VISON

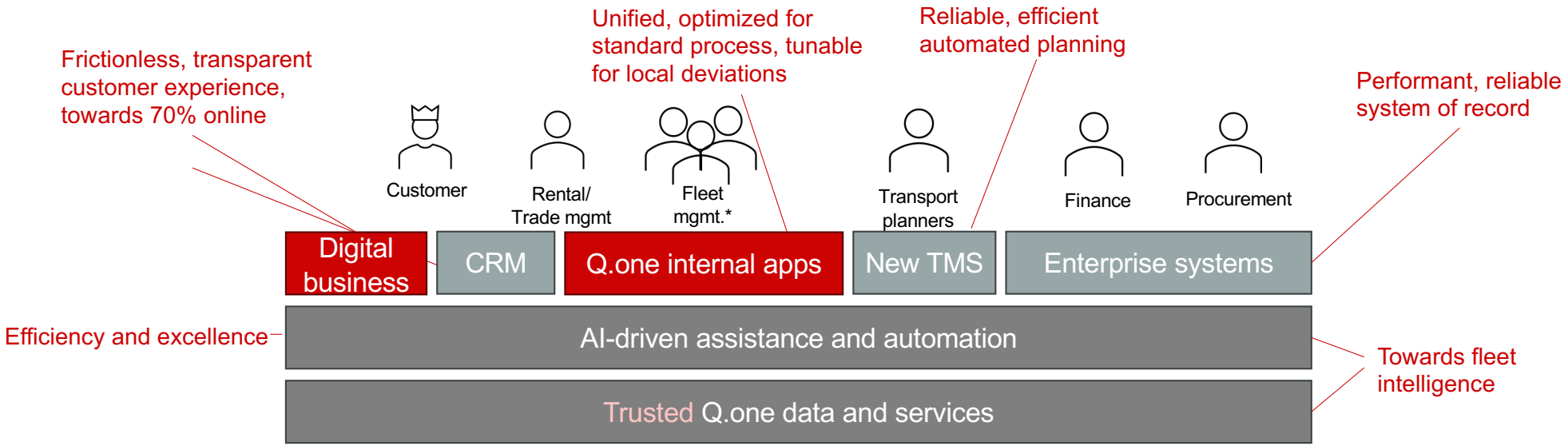
Product vision Q.one (to be refined)

Differentiating -> build

Non-differentiating -> buy

Buy and build

Get the machine you need
fast, easy, and ready to work



*drivers, technicians, machine planners, …

01

The Product Lifecycle

Goal

- Invest in products that align with business strategy and ensure future growth and returns
- Facilitate decision making on whether an initiative should (continue to) be funded

Guiding principles

Products don't exist in isolation. They sit at the intersection of:

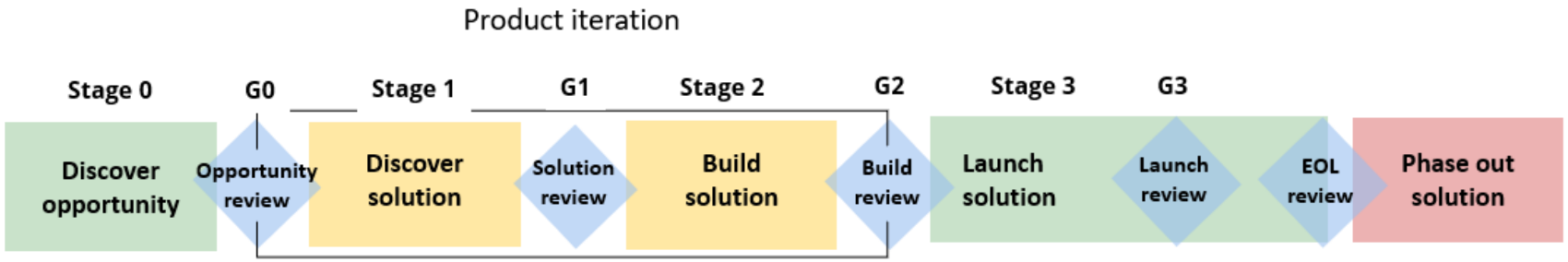
- **What the market needs** (customer problems and market dynamics)
- **What you can build** (your capabilities and solution)
- **How you operate** (your organization and go-to-market approach)

02

The Product Lifecycle gate model

THE PRODUCT LIFECYCLE GATE MODEL

Stage-gate flow

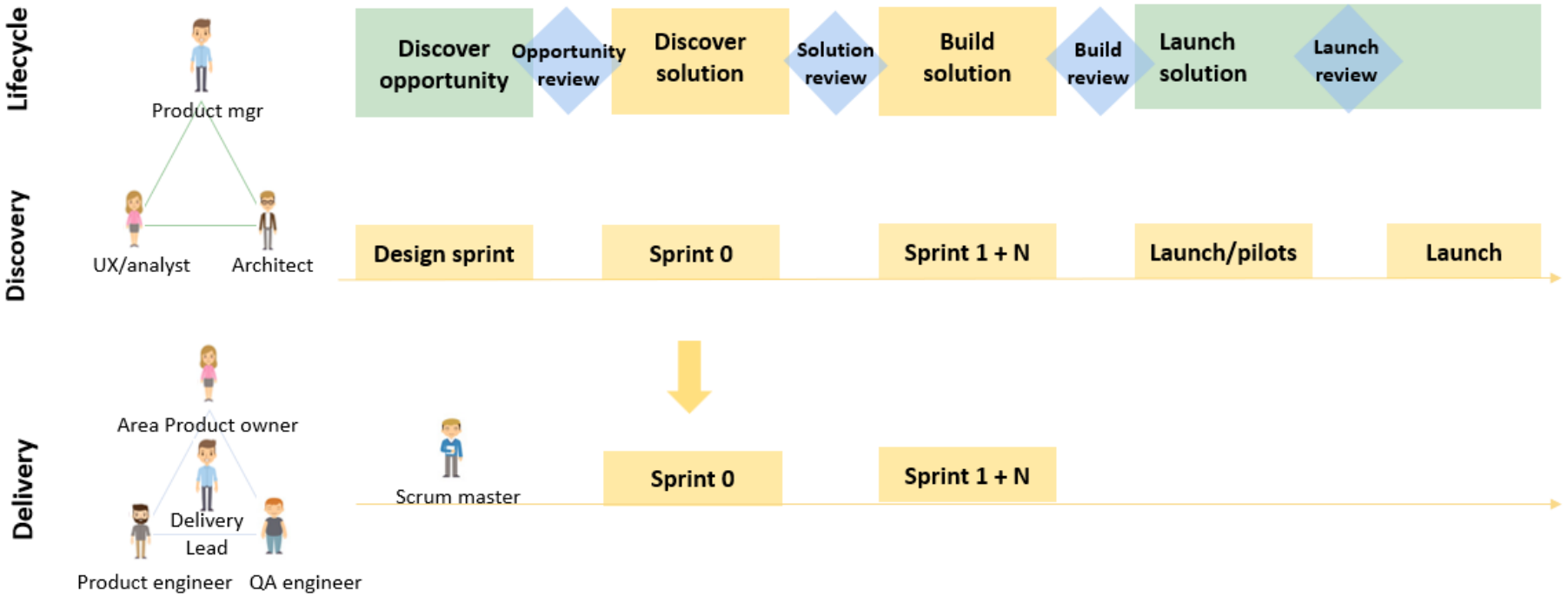


Key Takeaway

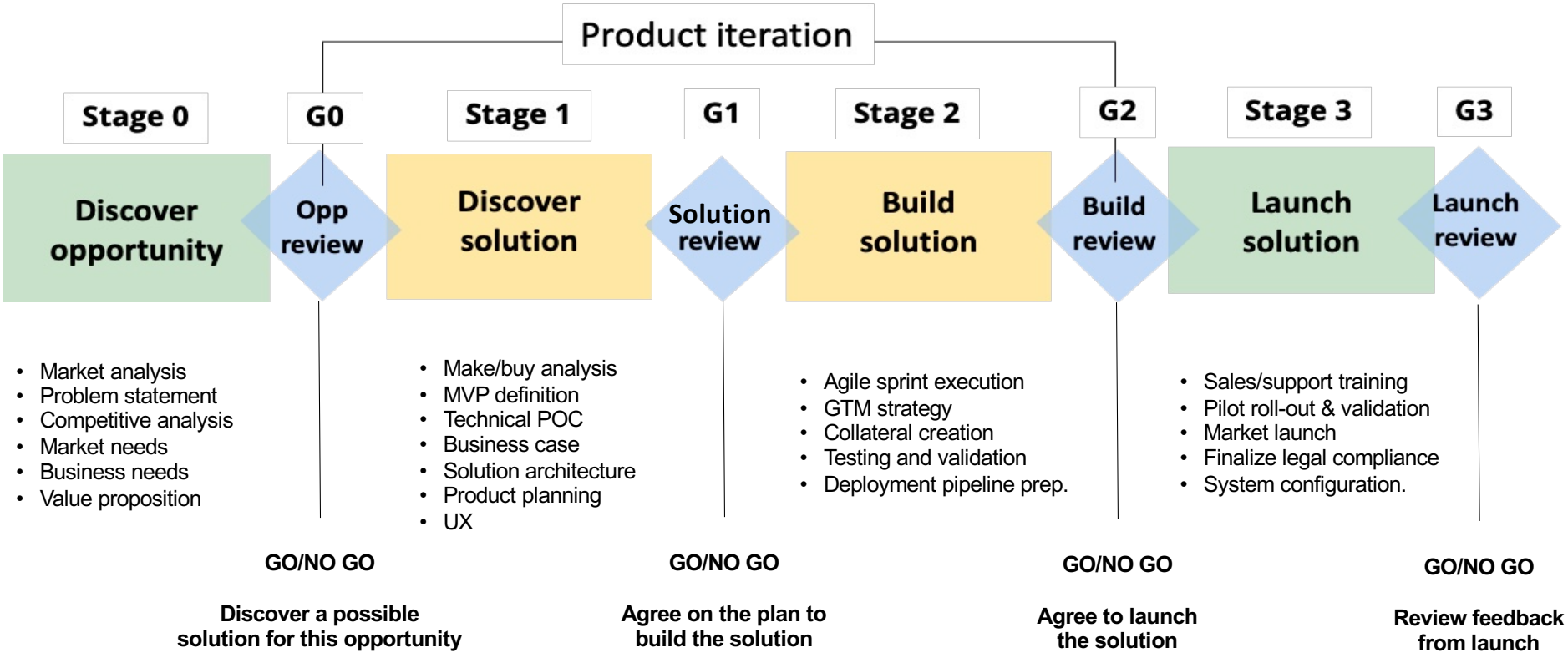
Each phase advances the product, with gates serving as critical checkpoints to ensure alignment and mitigate risk before moving forward.

THE PRODUCT LIFECYCLE GATE MODEL

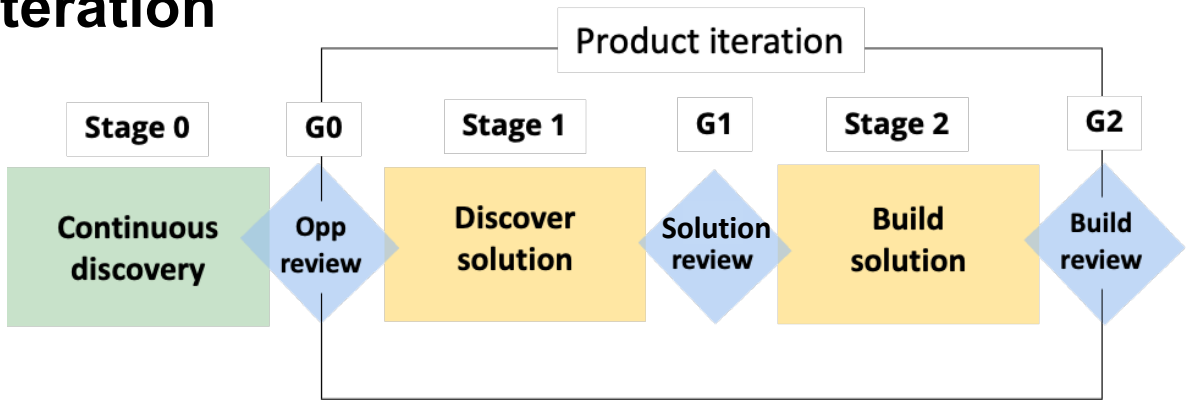
Stages & Gates



New product introduction



Product iteration



- Market analysis update
- Competitive analysis update
- Capture market needs
- Capture business needs
- Groom requests
- Roadmap update

- Product backlog update
- Update solution architecture
- Product iteration planning
- MoSCoW

- Agile sprint execution
- E2E testing and validation
- E2E deployment update
- Product collateral update.

GO/NO GO

Discover a possible new product iteration

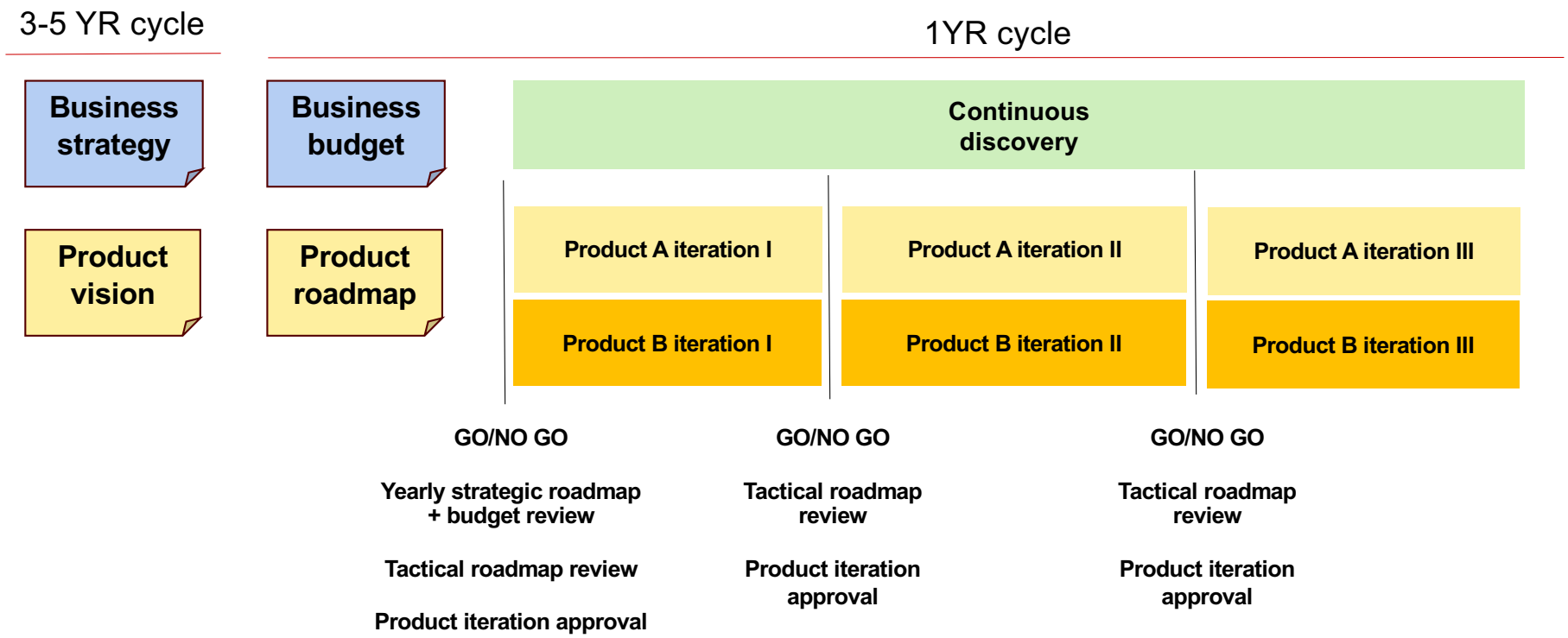
GO/NO GO

Agree on the plan to build the product iteration

GO/NO GO

Agree to launch the product iteration

Roadmap & iteration planning

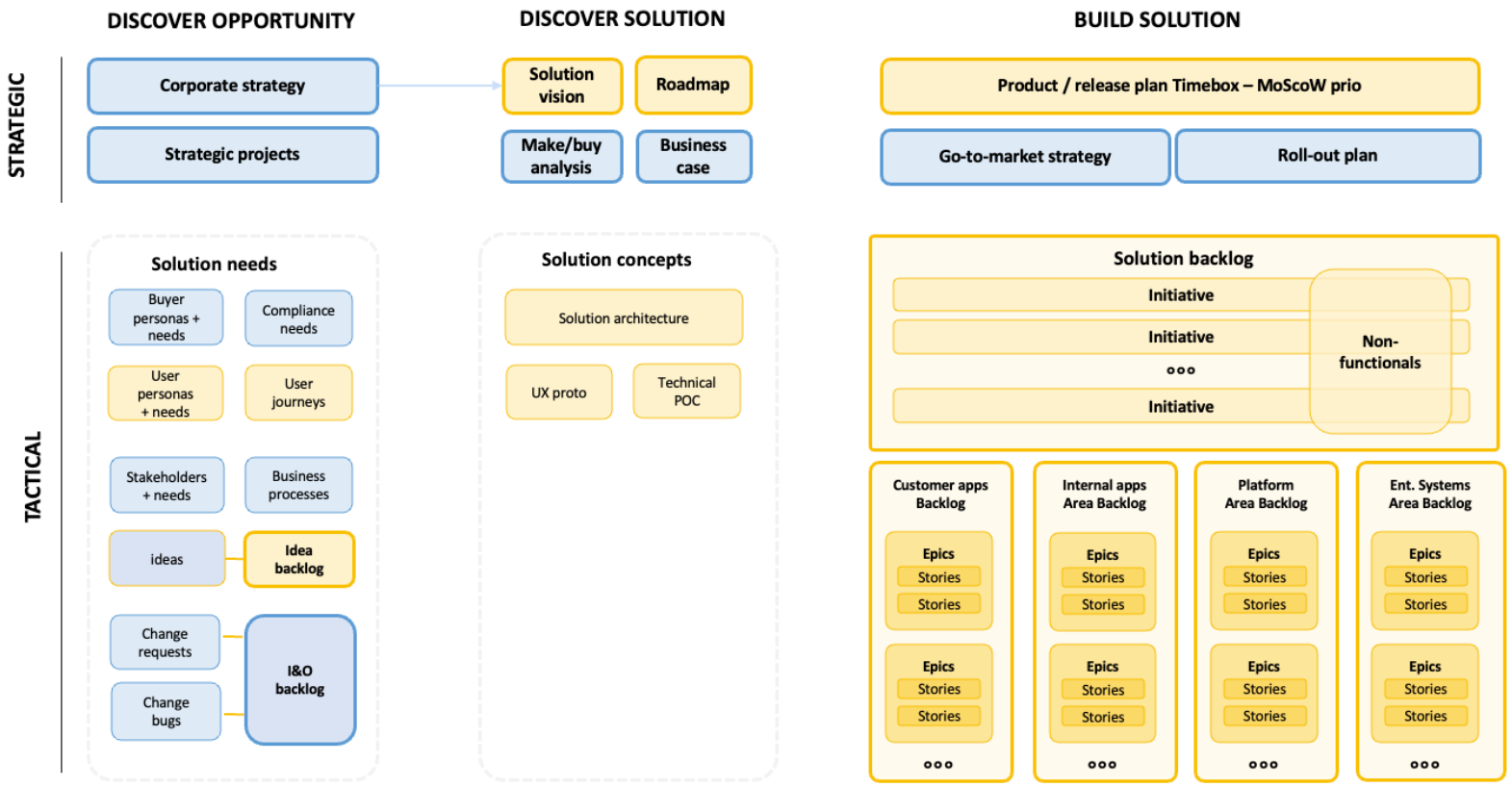


"Together, these processes ensure we're building products that matter, with the right level of governance without sacrificing speed."

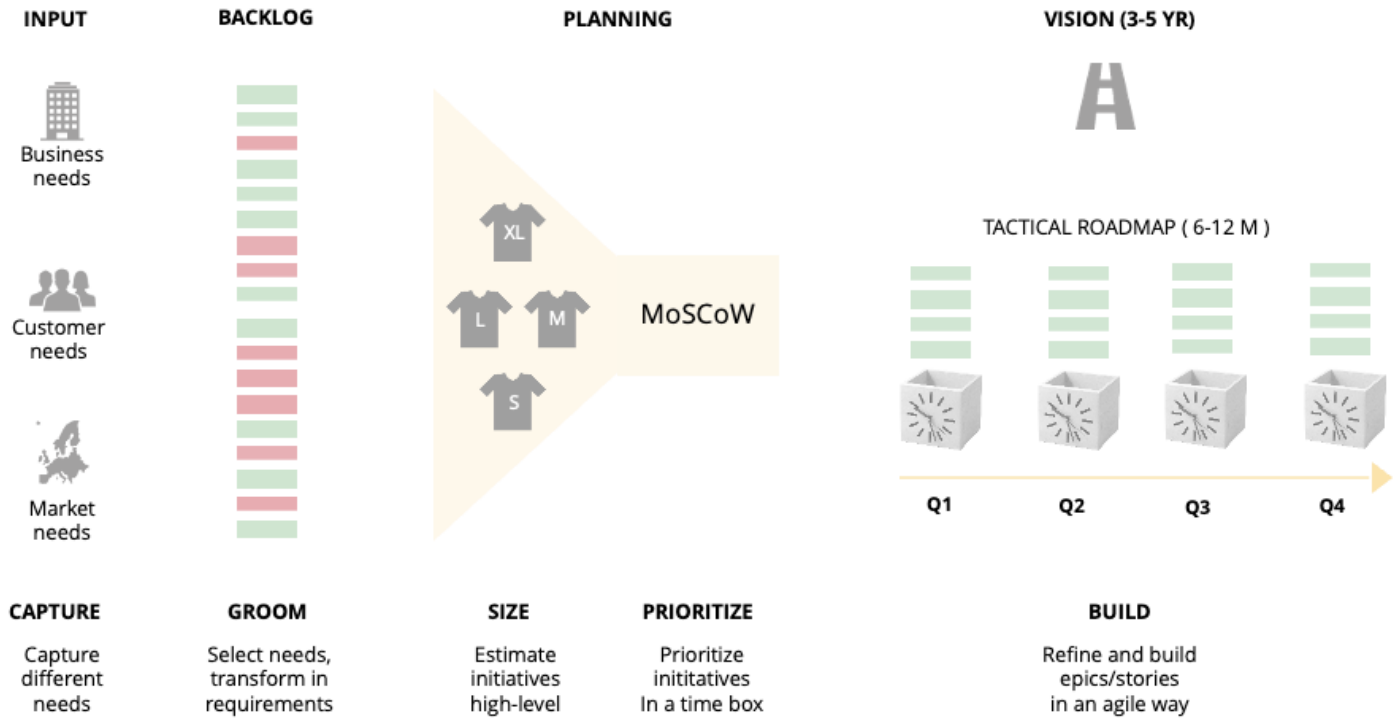
03

Requirements & Delivery model

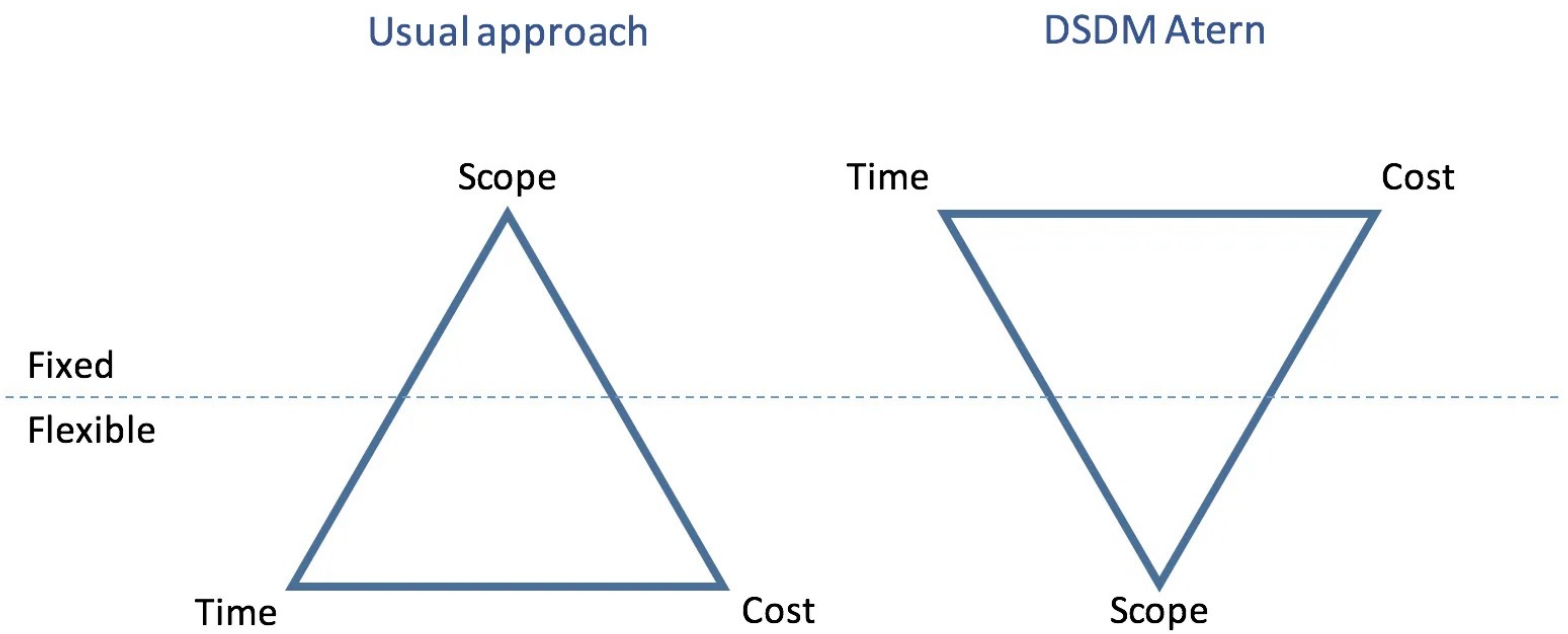
Requirements model



Timebox principles



The Iron Triangle



REQUIREMENTS & DELIVERY MODEL

MoSCoW Prioritization Framework

To bring clarity and discipline to our planning, we will use the MoSCoW method for prioritization. This simple yet powerful framework helps us negotiate scope and ensure that we focus our efforts on what truly matters for each delivery timebox.

Must Have

Non-negotiable requirements. The release is a failure without them. These are often legal, security, or critical path features.

Should Have

Important requirements that are not critical. They add significant value, but the release can still succeed without them. A workaround may exist.

Could Have

Desirable but less important. These are "nice-to-have" features that will be included only if time and resources permit.

Would Have

Features that are acknowledged as being out of scope for the current timebox but could be included in future releases.

Key Principle: By fixing delivery dates (timeboxes) and using MoSCoW to allow scope to be dynamic, we protect our deadlines and ensure we always deliver the most critical value.

REQUIREMENTS & DELIVERY MODEL

Key Success Factors for Timebox Delivery



Stakeholder alignment

The framework forces difficult conversations early about what's truly critical versus what's merely desirable. This prevents last-minute disputes about scope cuts and ensures everyone understands the trade-offs.



Progressive delivery

MoSCoW enables teams to deliver working software incrementally. You can ship with just the "Must haves" and add "Should haves" in subsequent releases, ensuring business value is realized quickly rather than waiting for a complete feature set.



Risk mitigation

By identifying and protecting critical requirements upfront, teams reduce the risk of delivering something unusable. The clear hierarchy means that if problems arise, there's a predetermined plan for what to sacrifice.



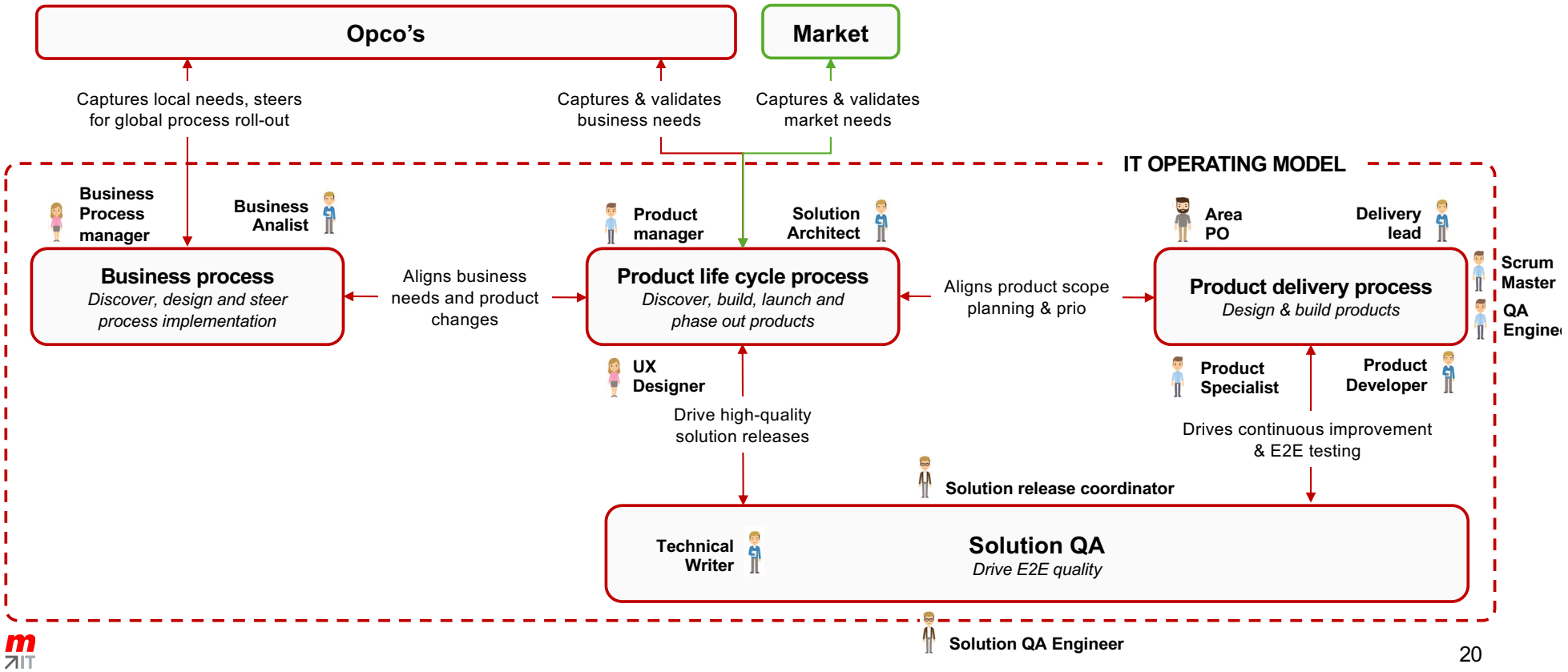
Realistic planning

The technique forces honest assessment of what can realistically be achieved within constraints. Teams can't simply say "everything is high priority" - they must make hard choices about relative importance.

04

Roles & Responsibilities

Simplified IT operating model

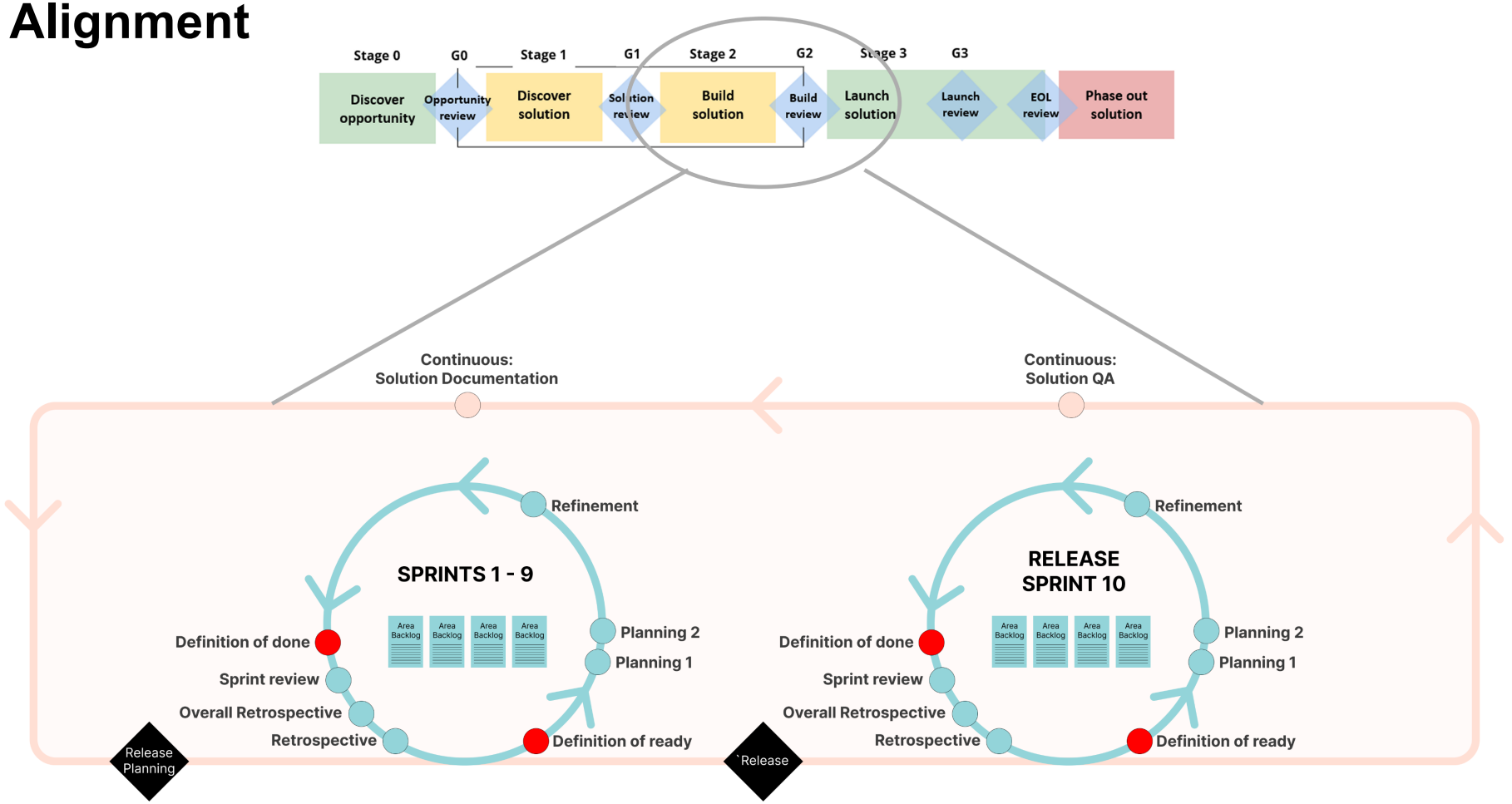


05

Aligning Product Lifecycle with Product Delivery processes

ALIGNING PRODUCT LIFECYCLE WITH PRODUCT DELIVERY PROCESSES

Alignment



ALIGNING PRODUCT LIFECYCLE WITH PRODUCT DELIVERY PROCESSES

Detailed Product Delivery Process

Sprint Planning One

Purpose: Product Manager presents prioritized items. Teams collectively select and distribute work, then define an overarching sprint goal.

Participants: Product Manager, All Scrum Teams, Scrum Masters, Area POs.

Key Deliverables: Sprint Backlog items assigned, clear Sprint Goal, cross-team coordination plan.

Refinement

Purpose: Teams work on detailing selected backlog items, addressing impediments and dependencies to prepare for the next sprint(s).

Participants: Scrum Teams.

Key Deliverables: Refined stories ready for development.

Sprint Review

Purpose: Inspect the Increment and adapt the Product Backlog if needed. Stakeholders collaborate on what was done and what to do next.

Participants: Scrum Team, Product Manager, Area POs, Stakeholders.

Key Deliverables: Feedback on the Increment, revised Product Backlog, insights for the next sprint.

Sprint Planning Two

Purpose: Teams break down selected backlog items into smaller tasks, confirm individual commitments, and finalize their sprint backlog.

Participants: Individual Scrum Teams, Scrum Masters, Area POs (as needed).

Key Deliverables: Detailed tasks for Sprint items, team commitment to sprint goals, updated task board.

Daily Scrums

Purpose: To inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work.

Participants: Scrum Team.

Key Deliverables: Daily plan, identification of impediments, updated progress.

Retrospectives

Purpose: Reflect on the sprint process and identify improvements for future sprints.

Participants: Scrum Team.

Key Deliverables: Action items for process improvement, team insights and learnings.

Questions?

Link to details on Confluence