/thoughtworks

Enterprise architecture playbook

A guide to enterprise architecture consulting

Table of contents

Table of contents

This playbook will continue to evolve, with new plays added over time (indicated by lighter font)

What is Enterprise Architecture?

The term “Enterprise Architecture” means many different things to people. We’ll take one rather narrow definition for the purposes of this book because it is intended to help Thoughtworkers deliver a specific type of consulting engagement.

For our purposes, Enterprise Architecture is the practice of aligning the people, processes and assets of technology to best deliver business outcomes.

Sometimes it can be described simply as aligning technology strategy to business strategy. It is important to note Enterprise Architecture almost always requires alignment between business and technology strategies, or implementing technology to deliver business outcomes. As is implied by the name, it is partly about architecting the entire enterprise, and shouldn’t be solely concerned with technology decisions.

In The Software Architect Elevator, Gregor Hohpe describes enterprise architecture as “the glue between business architecture and IT architecture” He breaks it down into the following 8 steps:

1. Understand the business strategy
2. Translate into an IT strategy
3. Create transparency
4. Define IT target picture
5. Define a roadmap
6. Harmonize and govern
7. Obtain feedback and refine
8. Coach and mentor

In this playbook we won’t cover any of the existing frameworks that have been codified over the years. Frameworks such as TOGAF and Zachman can be useful for remembering all the bits and pieces of Enterprise Architecture that must be defined and might be useful as an adjunct to the material covered here. But that’s not the main focus of our work in these consulting engagements.

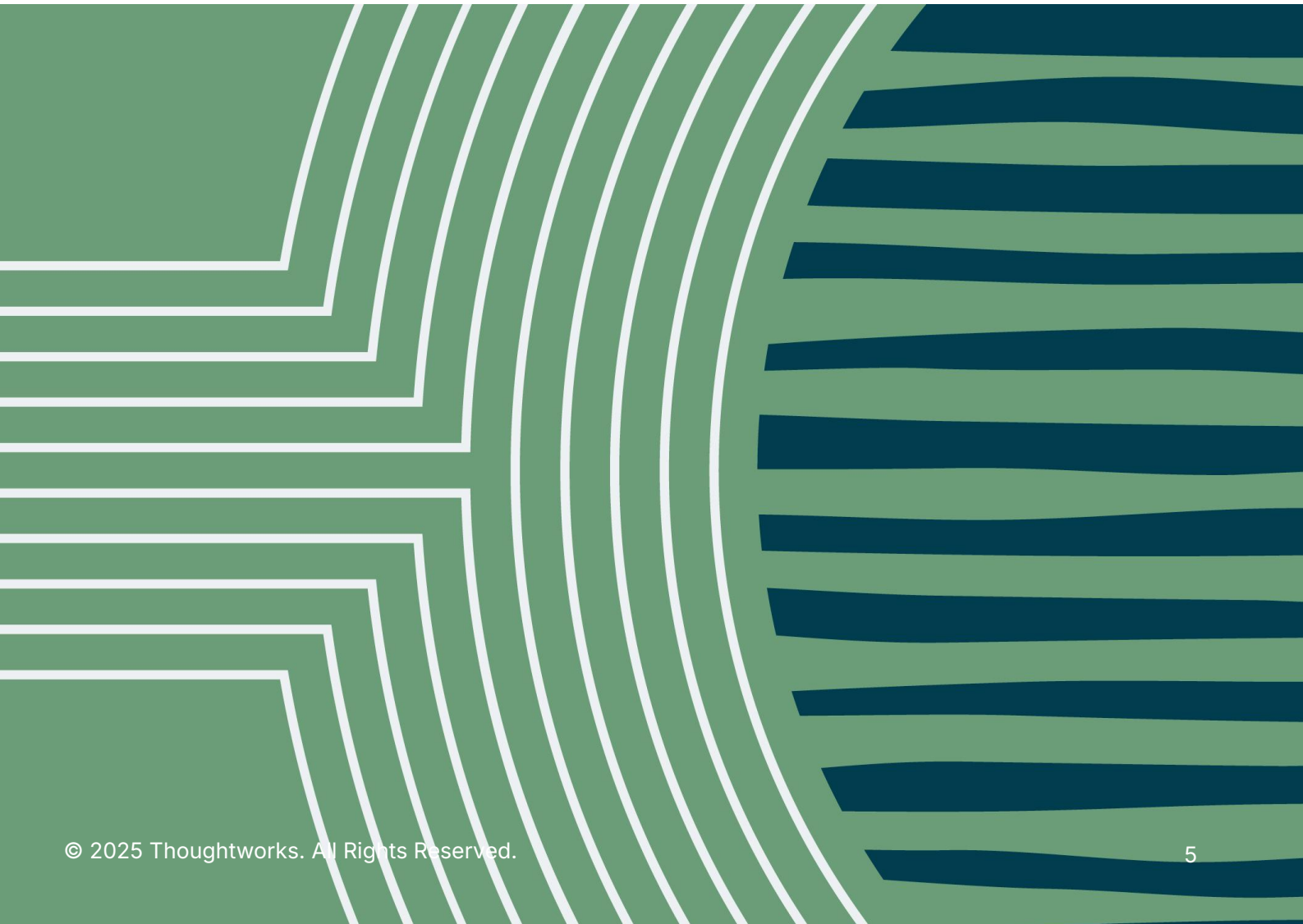
Introduction

This material draws on our experiences helping existing architects and enterprise architecture teams adapt their work to a modern digital business where businesses take a product-centric, test-and-learn approach to delivering software. Traditional enterprise architecture manages risk by requiring toll-gates and multiple stages of design before delivery can start. Our approach encourages continuous feature delivery and welcomes the inevitable changes in business direction that occur in today's marketplace.

This document should be seen as a toolkit of possible activities that can be used toward the purpose described above. The activities fall into 2 categories:

- Business Technology Alignment
- Execution and Governance

You should pick and choose the activities best suited to the organization, the architects themselves, and the current level of fluency that the organization has with digital business overall. There is no single best approach or blueprint. Every engagement will be slightly different and require a different set of tools to get the job done.



How to use this playbook

This playbook has been divided into two broad categories, **Business Technology Alignment** and **Execution and Governance.**

Business Technology Alignment

Business Technology Alignment looks to build an understanding of the business and of technology. It helps the Enterprise Architecture function and technology leaders understand:

- who are the organization's customers and partners
- what are the product, business and technology capabilities
- how does the organization deliver value and generate revenue
- what are the business and technology constraints the organization is operating within, and
- how can the organization align the business strategy with technology.

Execution and Governance

Execution and Governance seeks to understand how the Enterprise Architecture function can help the organization deliver value and generate revenue. Execution and Governance focuses on laying paved roads, and creating aligned autonomy for technology teams.

A **play** is a step-by-step guide to get you to a desired end state. It could be a stand-alone play, or one that is a stepping stone to another.

Plays don’t have to be executed in any specific order. Mix and match, and use the most appropriate play to solve your problem at hand.

Activities in this playbook

Business Technology Alignment

1. Organizational context
2. Technical architecture context
3. Data flow mapping
4. Business strategy
5. KYTE: Know Your Tech Estate
6. Stakeholder mapping
7. Stakeholder impact mapping
8. Future, backwards (Tech Visioning)
9. Operating model: the Architecture Group
10. Enable effective decisions
11. Lightweight architecture decision records
12. Artifacts on a page
13. Responsibilities matrix (RACI)
14. Operating model on a page
15. Capability mapping
16. Capability assessment
17. Capability heat map
18. Capability gap analysis
19. Wardley mapping
20. Feasibility analysis
21. Architecture roadmap planning
22. Systems health assessment
23. Emergent capability mapping
24. Business platform strategy
25. API strategy
26. Optimize outcomes: align business and technology

Execution and Governance

27. Path to production
28. Execution strategy
29. Fitness functions
30. Experiment tracking
31. Automated control at the point of change
32. Event storming
33. Domain driven design
34. Tech radar
35. Decision making framework
36. Architecture in the open
37. Architecture principles
38. Architecture diagrams and styles
39. RFC (request for comment)
40. Technical debt tracking
41. Paying off technical debt
42. Architecture workload management
43. Architecture advisory forum
44. Talent growth assessment

This playbook will continue to evolve, with new plays added over time (indicated by lighter font)

Business Technology Alignment

Play #1

Organizational context

Understand the structure of the organization, their product space and their technology landscape.

Understand what the business is, how they are structured, who their customers are, what value they deliver and how they deliver that value.

Use this play to:

- Identify your stakeholders
- Understand the space you're operating in within the organization
- Provide a quick assessment of organizational complexity
- Understand the organization's product offering and identify who their customers are

Expert tip

It's critical you have an understanding of the business, their customers and how they deliver value. This, along with Play #2 - Architectural Context, and Play #4 - Business Strategy will set you up for success.

Time required

1 - 2 days

Who participates

Project team

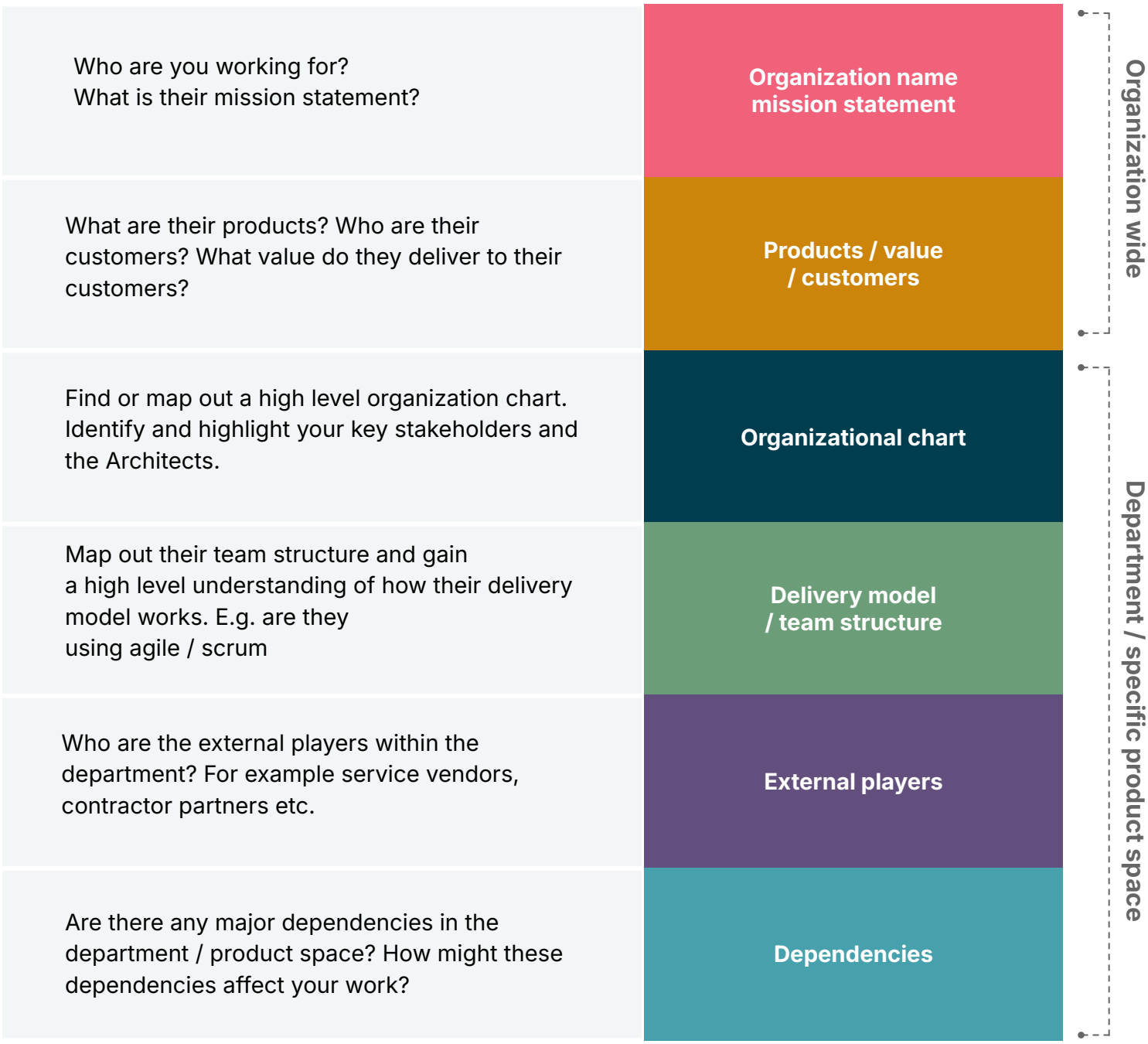
How often

Once-off / as needed

Example

Organizational context

First start by understanding (at a high level) who the organization is, their mission statement and what their core products or services are. Once you've got this picture, drill into more specific details on the space you're working in. This could be within a department or product.



Play #2

Technical architecture context

Like business context, having technical architecture context is crucial for successful Enterprise Architecture initiatives.

Architecture context provides the foundation for making informed decisions, identifying and mitigating potential risks, and ensuring the architecture effectively serves the business needs in the long run.

Whilst gathering architecture context, it is important to wear a business hat as well as a technical hat and ensure you understand the architecture within the relevant business/domain context. This includes understanding how regulatory policies or internal policies impact the architecture too.

Use this play to:

- Identify and understand the current technical architecture
- Provide a quick assessment of architectural complexity
- Understand how the technical architecture supports core business functions

Expert tip

Beyond simply diving into the technical architecture, understanding how the architecture supports core business functions is essential. To do this, follow the money. Engage with business stakeholders to follow the financial flow to see how technology directly impacts revenue generation, cost reduction, or other key metrics.

Time required

1 - 5 days

Who participates

Architects, project team

How often

Once-off / as needed

Technical architecture context

1. Bound your context

Before you start, determine the required breadth level of detail. Do you need fine detail? Or will a level 1 systems context view suffice? By clearly defining the context, you can ensure that your research and analysis is focused, relevant, and effective.

2. Desk research

Gather as much context as you can through existing documentation. See what is discoverable by searching the organizations wiki and shared drives. In doing so, you will also gauge their practices around documentation, communication, collaboration and transparency.

3. Interviews and workshops

Have architects, principal or lead engineers walk you through their architecture diagrams. If diagrams don't exist, draw it with them. You may need to engage with many folks across the organization to draw the existing architecture before you get to a holistic picture. You don't need to make it a formal drawing, keep it simple with boxes and lines.

In some cases, where knowledge has left the building, or for complex systems, running an event storming workshop or path to production workshop might help uncover critical aspects of the architecture.

4. Broader context

Reach further into the tentacles of the organization to get a broader understanding of the architecture, how it supports and enables the business, and what regulatory/internal policies impact the architecture. Importantly, follow the financial flow to see how technology directly impacts revenue generation, cost reduction, or other key metrics.

5. Trust but verify

Validate stakeholder input by selectively investigating code, configs, or testing, focusing on key assumptions rather than a full assessment. Define a bounded scope using sponsor insights and supplement with production metrics or SLA/SLO data to surface hidden risks.

Play #3

Data flow mapping

Understand the structure of the system, the types of data used and the major components and their responsibilities.

Use this play to:

- Identify the major components of the system and their responsibilities
- Identify the main data flows of the system
- Understand main integration points
- Understand main users of the system

Expert tip

Most systems will have more than one major data flow. It's often clearer to show each flow on a separate diagram but use the same structure and positioning of components across all of them to aid the team in understanding how the flows relate to one another.

Time required

30 - 60 minutes each

Who participates

Architects, principal developers, data analysts

How often

As needed

Data flow mapping

At the end of this play, you should have one or more data flow maps that show how data moves from one user or system to another. It's useful to have a high-level understanding of the major components before beginning this play.

In order to run this play successfully, you'll need access to people familiar with the systems involved, as well as either a physical or virtual whiteboard.

1. **Identify the major data flows.** For example, for a product catalog for an e-commerce retailer, these might be adding a new product to the catalog, updating the images for an existing product, and retiring a product from sale.

2. **Identify the major users and systems involved.** See the following page for an example.

   If you're running this play physically, use a single sticky note for each user or system.

3. **For each of the identified data flows, start at the origin and trace its path through each system until completion.**

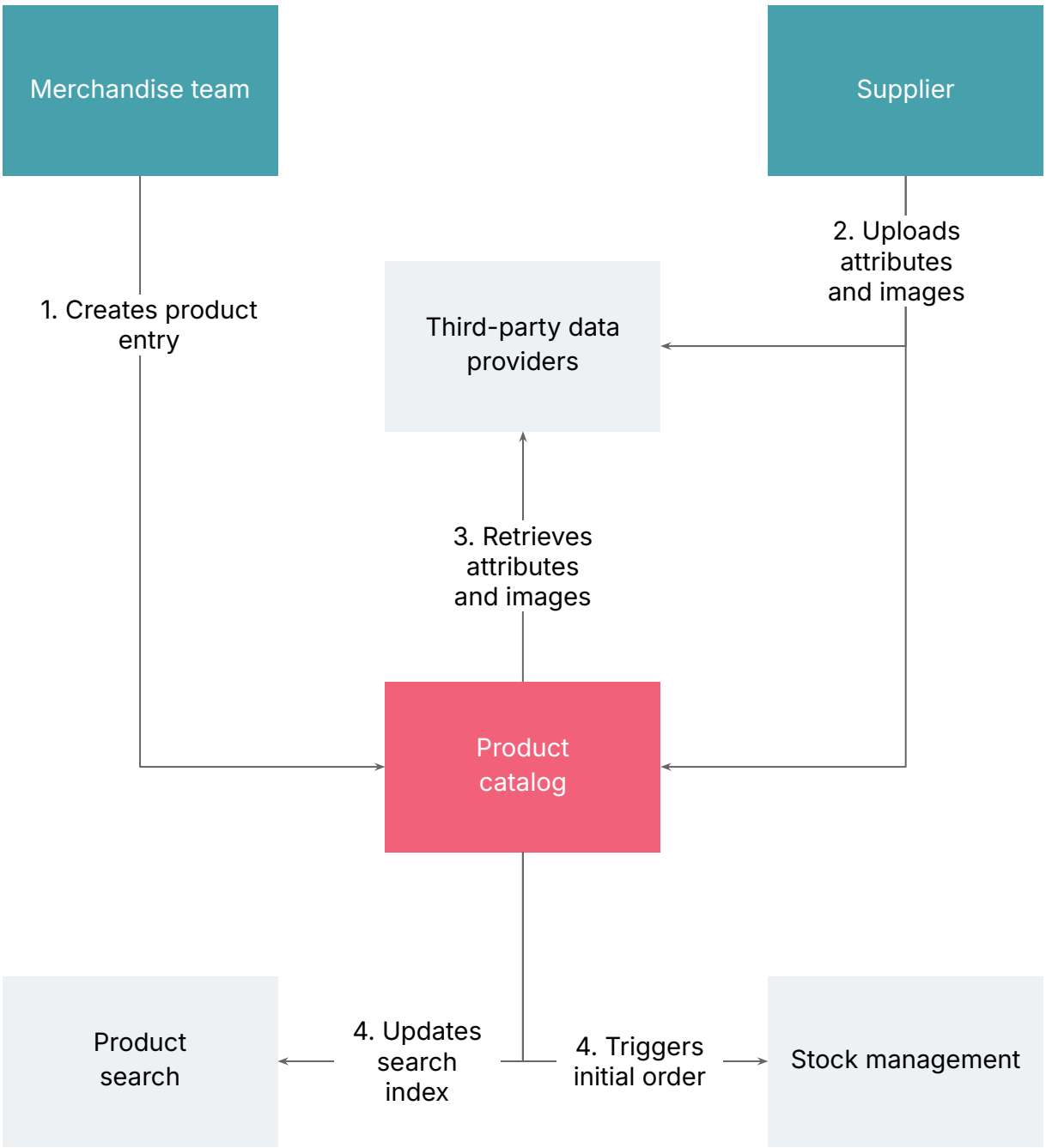
   Number and annotate each step in the flow to explain the series of steps. For example: "step 2: retrieve product images from third-party data provider".

   If you're running this play physically and are drawing multiple flows on a single diagram, it can be helpful to use a different color for each flow to distinguish them from each other. If you're running this play virtually, it's usually clearest to create a separate diagram for each flow.

Example

Data flow mapping

This is an example data flow map for adding a new product to a product catalog system for a fictional e-commerce retailer:



Play #4

Business strategy

One of the very first things an Enterprise Architect must do is understand the business strategy, along with the business goals and measures of success.

Understanding the business strategy is not as simple as having a copy of the strategy shared with you. It is important to know everything that played out behind the scenes and the thinking that went into creating the strategy. Whether a business strategy, goals, objectives and measures of success exist or not, this play will help you discover what exists and why, and how it is perceived in the organization.

Use this play to:

- Understand the business strategy
- Understand the business goals and objectives and how these are measured
- Understand how the business strategy came about and how it is communicated to the organization

Expert tip

If a business strategy does not exist or is hard to pull out of individuals, try to reverse engineer one by researching annual general reports and media releases. Once you've constructed something, playback for key stakeholders to correct. Sometimes presenting something wrong can elicit what you're after.

Time required

1 - 3 hours

Who participates

Architects, executives, key senior stakeholders

How often

Once-off

Business strategy

	is...	is not...
Purpose	• the reason to exist • why the organization exists beyond making profit • motivational • a statement that speaks to all	• about profit • directed at a small group of employees
Vision	• aspiration for the future • the north star • a destination • direction	• a near-term goal
Business strategy	• informed by the vision • alignment of opportunities, plans and behaviour • an actionable plan to realize the vision • accessible • the how	• an idea of the future independent of the vision • a list of goals • no actions • inaccessible
Goals	• actionable stepping stones derived from the strategy • time sensitive • measurable • considered and well thought out • strategic objectives	• unrealistic • everything all at once • processes • ad hoc

Business strategy

1	**Identify stakeholders to walk you through the strategy**	If the person responsible for the business strategy is easy to identify, excellent. If not, seek to meet with executives and senior business leaders to take you through the business strategy, goals and objectives and measures of success. Hearing it from different stakeholders will highlight how well the strategy is understood at the top of the organization and where there might be misalignment.
2	**Walk through of the strategy**	Once you have time with key business leaders and executives, ask them to walk you through the strategy. It's important to listen first. Allow the stakeholders to share the strategy/goals/objectives/measures of success first, before diving into questions. Interrupting and jumping into questions might make the stakeholder/s feel like you are judging the strategy and they may become defensive.
3	**Gain a deeper understanding**	Use the list of questions on the next page to get a deeper understanding of why the strategy exists, how it was/is communicated to the organization and how well they are executing the strategy.
4	**Playback your understanding**	Take some time after the initial meeting/s to gather your notes/collective thoughts, then book time to playback your understanding of the strategy with those stakeholders who shared it with you. In doing so, you gain clarity on your understanding of the business strategy and and correct where you may have misinterpreted the strategy.

Business strategy

Questions to gain a deeper understanding of the business strategy.

- How did the strategy come about? i.e who was involved in the creation?
- When was the strategy introduced?
- When was the last time this strategy was reviewed?/How often is the strategy reviewed?
- How was/is the strategy shared with the organization?
- How can teams access the strategy? Where is it published?
- Do you feel it is well known and understood across the organization?
- What is the impact to the business to execute the strategy?
- How well is the organization executing the strategy?
- What is helping the organization execute the strategy?
- What is holding the organization back from executing the strategy?

Questions to gain a deeper understanding of the business goals and measures of success.

- What are the business goals and measures of success
- How do they align to the business strategy?
- How well is the organization tracking towards these goals?
- What is helping the business excel?
- What is preventing the business from delivering on their goals and realising the vision?
- How frequently are the goals reviewed?
- How often are progress updates shared with the organization?
- How do teams associate their work to the business goals?
- Who is responsible for the business goals?
- How are the goals and measures of success determined?

Play #5

KYTE: Know Your Technology Estate

Once you have an understanding of the architecture, the next step is to capture all of the components and resources associated with the architecture to create a comprehensive picture of the technology estate.

Without a comprehensive understanding of the technology estate, blind spots may arise, leading to inefficiencies, security vulnerabilities, and missed opportunities.

By understanding the technology estate, you can ensure it aligns with business goals, optimize costs, improve security posture, and make informed decisions about future technology investments.

Use this play to:

- Create a detailed record of your organization's architecture components (e.g. a deployable unit) and resources
- Identify ownership responsibilities and any ownership gaps
- Understand how the architecture is supported and maintained across the organization

Note: this play assumes the organization lacks a comprehensive service repository or internal developer portal documenting its technology landscape.

Expert tip

Make this a collaborative effort across technology. Be transparent in the purpose and benefits and how folks can contribute. Share the findings and insights and open up a discussion with the technology organization to help identify areas for improvement and gain buy-in.

Time required

1 - 4 weeks

Who participates

Architects, delivery and technology teams and team members

How often

Once-off, then ongoing maintenance as required

KYTE: Know Your Technology Estate

Data storage and management

Before getting started decide how the information will be collected, stored and managed. For example, will you be using a spreadsheet or perhaps a CMDB (configuration management database) tool?

Who will have access to it and how will you ensure it is secure?

Potential challenges

You may face resistance from teams. Clearly communicate the purpose, what you need, and the benefits to them and the organization.

Have a technology leader set context in a wider forum and reinforce it through management lines. This ensures teams are aware before you reach out.

Minimize their burden - opt for a 30-minute conversation with them instead of sending out a survey.

Ongoing maintenance and updates

Once you have collected the information, ensure you have a process and mechanism in place to keep the information up to date.

This might mean having ‘champions’ responsible for maintaining it, it might mean as a group of Architects you take on responsibility. However it looks, make sure you’ve established responsibility and a cadence to maintain the inventory and a mechanism for updates.

Gathering information

This play describes collecting information by asking questions and conversing with teams. In addition to this, automated discovery tools can also identify and capture services and configurations. These tools are especially valuable when knowledge gaps exist.

Consider leveraging observability tooling, security scanners, or cloud inventory solutions to streamline this process.

KYTE: Know Your Technology Estate

1	**Determine the scope of your inventory**	Decide what information you are looking to collect. Given you will be talking to all teams, it could be beneficial going a little broader in your questions to gain a comprehensive understanding of the technology landscape within the organization. For example, in addition to recording the services teams maintain and the technology stack they are using, you might consider collecting information on their engineering and deployment practices too.
2	**Draft a set of questions and possible responses**	Draft a set of questions and responses (note that you will want to keep responses as consistent as possible for record keeping and comparison, hence drafting a standard set of responses for each question).
3	**Sense check**	Sense check the questions with a couple of principal or lead engineers, or engineering managers.
4	**Identify teams**	Gather a list of all delivery/development teams. Depending on the organization and how long you've been there this could be a quick and simple task, or a drawn out one. Include all teams that build and deliver products, be it internal facing products (i.e. a platform engineering team), data products, or external facing products.

KYTE: Know Your Technology Estate

5	**Identify delegates within each team**	Identify 2 representatives from each team. You will want 2 developers of different seniority to get a well rounded perspective. We don't generally include the whole team as we find it can slow things down and may see unnecessary debates arise within the team. You do want folks who are close to the code. An engineering manager or principal engineer who is not hands-on coding day-to-day won't be the right people to speak with.
6	**Schedule time with each team**	Next, you'll want to schedule time with each team to go through the questions. Thirty minutes is generally enough time to go through all of the questions per team. In large organizations, group sessions for similar teams is an efficient approach. As tempting as it might be to simply send a survey and ask folks to complete it, we find having face-to-face conversations much more effective for a number of reasons, including: • they help you to build or extend your relationship with the team/developers • you gain additional context through the conversation that you wouldn't get via a form • you get better consistency in responses • both you and the team representatives can ask clarifying questions • you take the burden off the teams - have you ever heard of someone getting excited about a survey?

KYTE: Know Your Technology Estate

7	**Collect information from the teams**	Collect information from the teams. You may want to do a trial run with a couple of teams from different departments first and make any necessary adjustments to the questions before continuing on to all teams.
8	**Collate and share**	Once you've gathered all information collate it, visualise it and make it accessible. While the purpose of this exercise may have been for you and your team to deepen your knowledge of the current technology landscape, inventory components and the technology stack, the rest of the organization will also find the information insightful. Share what you have, run a walkthrough session and open up a discussion on what you are seeing.

Play #6

Stakeholder mapping

Knowing who your stakeholders are and how to communicate with them could be the difference between having a successful, visible architecture presence in the organization or architecture being an afterthought.

Stakeholder mapping is essential for the success of any team or project and allows teams to prioritize and target time spent building relationships and to identify risks in their stakeholder landscape.

Use this play to:

- Identify stakeholders
- Define communication methods and frequency of communications with stakeholders
- Better manage stakeholders expectations

Expert tip

Stakeholder mapping may not appear be a high priority for Architects, but getting this right by knowing who their stakeholders are, who needs to be influenced, and how to communicate with them will help put architects on the map within the organization and help them to deliver their work successfully.

Time required

90 minutes

Who participates

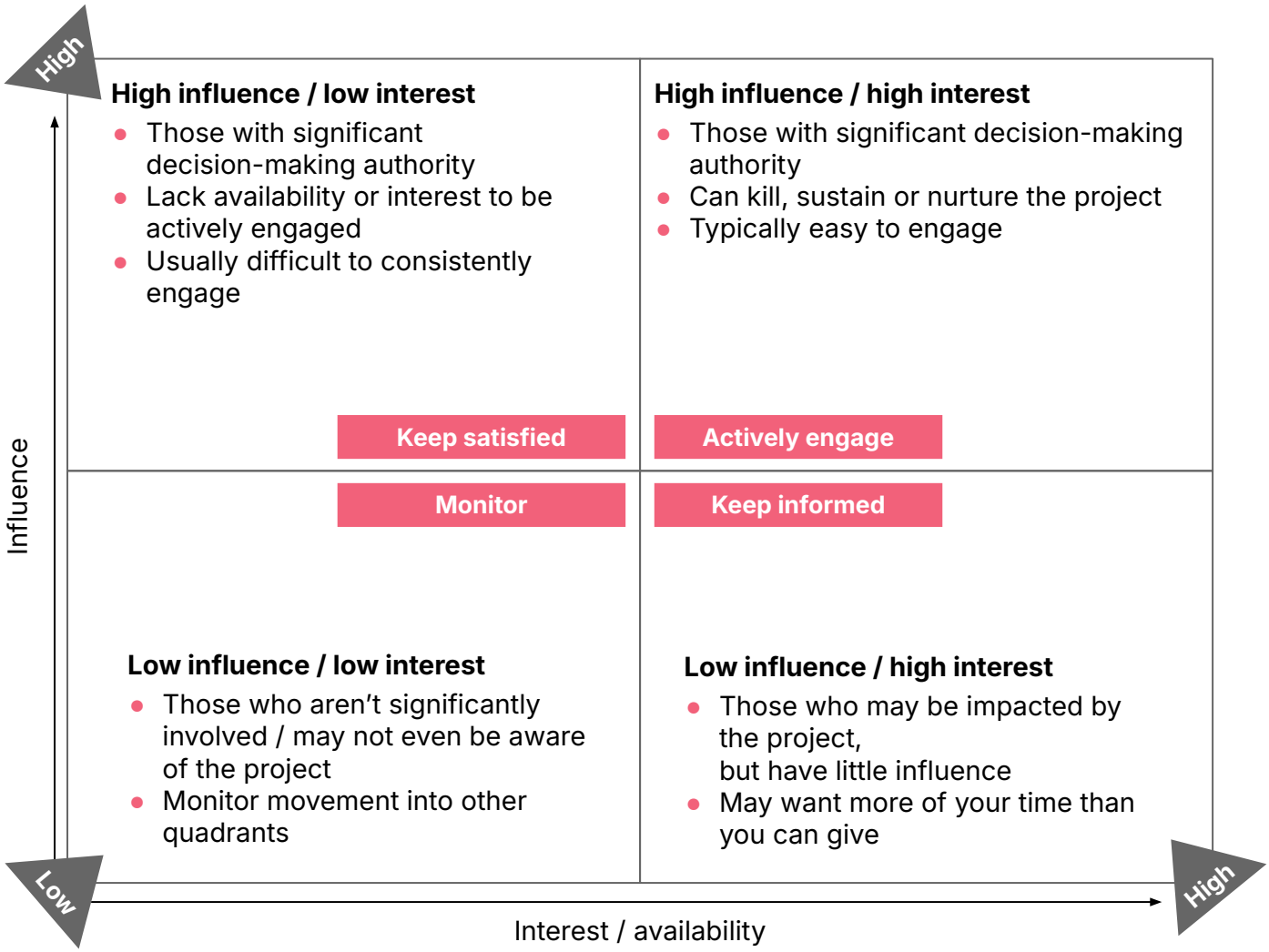
Architects

How often

Every 6 months

Stakeholder mapping

A stakeholder is an individual (or group of people) who have an interest in, or influence of, the Architects or the work the architecture team produces.



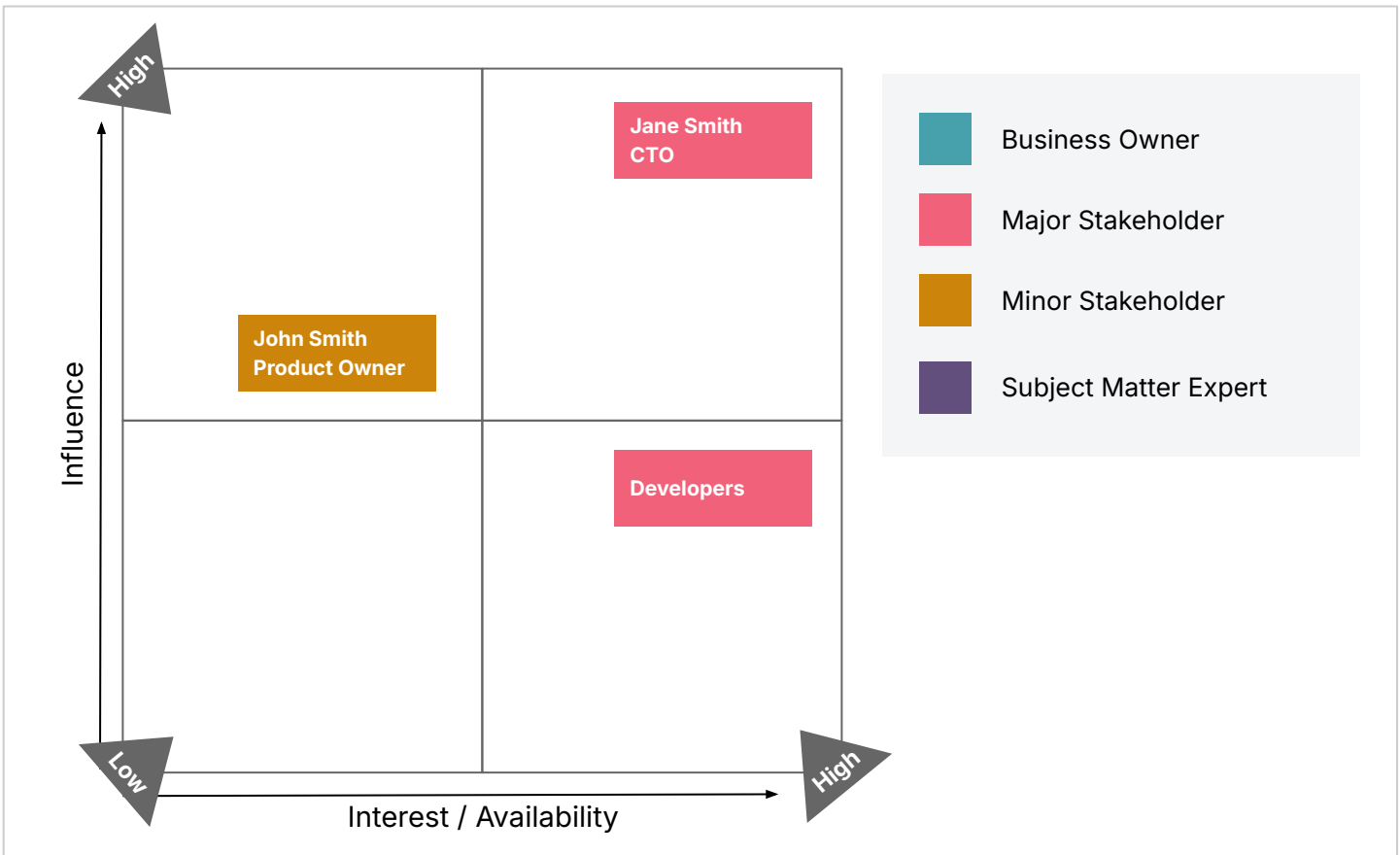
Publication

It's generally a good idea to keep publication of a stakeholder map internal to the team. Stakeholders may not like seeing themselves ranked as "low influence".

Related artifacts that are safer to make visible include a comms plan (for each stakeholder, how and when will we communicate / interact), and a RACI matrix (who will be involved for what kind of things, with what decision authority).

Stakeholder mapping

1. **Make a list of stakeholders**
   - On sticky notes, ask team members to write down the names and roles of their stakeholders (one stakeholder per note).
   - Use different colored post-its to represent internal vs external stakeholders.
2. **Prioritize**
   - Once you have defined the list of stakeholders, place them on the map according to their level of interest / influence. This step can involve a lot of discussion between team members so allow plenty of time for this.
3. **Categorise**
   - Using sticky dots, indicate if the stakeholders are a Business Owner; Major Stakeholder; Minor Stakeholder; Subject Matter Expert



Play #8

Future, backwards

As a group, create a shared understanding of the current architectural state and what events in the past led to this. Examine how the organization might navigate towards a more desired state and away from an undesired one.

Future, Backwards will be a step towards defining a technical vision. It's important to create a shared understanding of where the organization is at, and how it came to be. You will likely find, depending on tenure, the picture will look different for each person in the group. There is no right or wrong answer, record all perspectives as you go.

Use this play to:

- Embed lessons from the organization's past in decision making
- Aid in conflict resolution between different groups with opposing views
- Create a shared view of a complex problem
- Give leaders an overview of the hopes and fears of their organization
- Help leaders understand which entrained patterns of past perception are influencing the organization's future

Expert tip

This is a great activity to start forming the **technical vision** and desired to-be state.

Time required

2 hours

Who participates

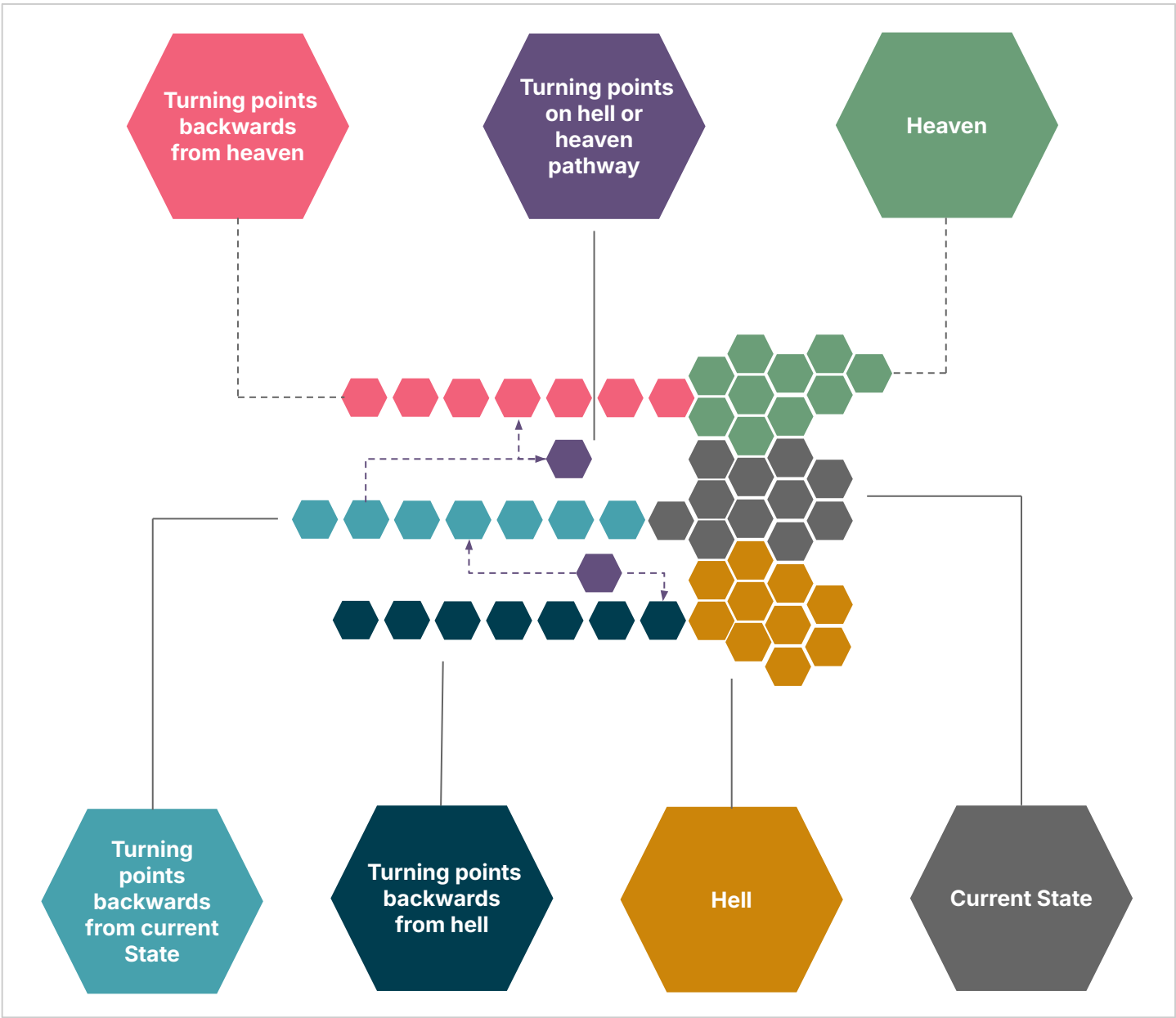
Architects

How often

Once-off

Future, backwards

The diagram below describes the 7 states that make up the activity. The picture shows the shape of the diagram and where the hexies from each state should be placed as you work through the activity.



Future, backwards

1	**Overview of future, backwards**	Provide an overview by letting participants know the activity allows us to take a complex problem and understand the many factors around it and to construct a vision of the system - it will help us to create a shared view of a complex problem. Let people know we will be working through (significant) ambiguity; **the process requires you to go through a series of steps without knowing the outcome until the process is finished.**
2	**Set the scene**	Run through how we'll work (groups, tooling, safe space). Run a short warm-up activity so people feel comfortable.
3	**Current state**	Ask the group to spend ~5 mins writing down as many points as they can to **describe the current state** (1 per hexie). For remote sessions, as the facilitator, be the scribe too to allow participants to maintain a conversation and stay on the same page. If you're running the activity with multiple groups, it would be best to have a facilitator within each group.
4	**Turning points backwards from current state**	What is the most significant event in the immediate past which shaped/caused the current state? Ask the group to identify the **turning points (key events) back from the current state.** Spend a maximum of 3 minutes per event for a total up-to 20 mins. It's natural for participants to name an event 2 years in the past, or to request to work from a point in time in the past, back to current state. To construct a vision of the system by its events that is not causal in nature and therefore not subject to narrative bias it's important to work backwards, from the current state.

Future, backwards

5	**Heaven**	What is the best possible scenario state in the future? Imagine an **impossibly** good future (heaven) and describe the conditions/experience of heaven. There are no constraints. Money is not an issue. People are not an issue. Technology is not an issue. Think big. Ask the group to spend ~5 mins writing down as many points as they can to describe heaven (1 per hexie).
6	**Turning points backwards from heaven**	Make heaven happen. Connect heaven to a past event with fictional events. What is the most significant recent event that preceded the heaven state? Ask the group to identify the (fictional) turning points (key events) back from heaven. Spend a maximum of 3 minutes per event for a total up-to 20 mins.
7	**Hell**	What is the worst possible scenario in the future? Imagine an **impossibly** bad future (hell) and describe the conditions/experience of hell. Ask the group to spend ~5 minutes writing down as many points as they can to describe hell (1 per hexie).
8	**Turning points backwards from hell**	Make hell happen. Connect hell to a past event with fictional events. What is the most significant recent event that preceded hell? Ask the group to identify the (fictional) turning points (key events) back from hell. Spend a maximum of 3 minutes per event for a total up-to 20 mins.

Future, backwards

9	**Turning points on heaven or hell pathway**	What are the key turning points describe in current state (backwards) that would help us to get on the path to heaven? or on the path to hell? What would make the difference between the heaven and hell paths at this point? Ask the group to identify 1 – 2 turning points. These are linked with arrows.
10	**Close**	If you've run the activity with multiple groups, bring them back together to share their outcomes. Wrap up the activity and let participants know next steps.

Source.

Play #9

Operating model: The architecture group

An operating model will help a group of Enterprise Architects find alignment and work towards a common goal.

The operating model looks at the business strategy and the architecture strategy to define an enterprise operating model enabling the Enterprise Architecture function to be embedded in the organization while delivering value.

There are four parts to an operating model:

1. The architecture group
2. Making decisions
3. Recording and sharing decisions
4. Responsibilities (RACI matrix)

This play will cover part one, the architecture group, looking at the group's structure and how it maps to the organization as a whole.

Use this play to:

- Understand how Architects work within the organization today, and how they would like to work in the future
- Create an operating model to meet the Architects' desired state
- Define how the Enterprise Architecture function will deliver value to the organization
- Build and sustain high performance

Expert tip

Architects can often work in a siloed capacity. Building out an Operating Model will enable them to realize their value and how they deliver it to the organization.

Time required

2 - 4 hours

Who participates

Architects

How often

Once-off, then revisit each time a new architect joins, or every 12 months

Operating model: The architecture group

1. Create a shared understanding of the business strategy. Get the architecture group together to pull apart, discuss and build a shared understanding of the Business Strategy.
2. Get clear on the value of the Enterprise Architecture function and how the function can help the business realize its vision. An example of how to start this is by having the group of architects brainstorm answers to these 3 questions:

What do we care about?	Why are we here?	What value do we add to the organization?
e.g. Architecture that matches the organization	e.g. Providing alignment between technology and the business	e.g. Technical guidance / decisions

3. Record how they currently operate. If the group of architects have been operating in silos, this could vary greatly. It's good to get it all down on paper to help create a shared understanding along the way. The questions below will help paint the picture. Once you've done this, understand their current cyclical meetings. Along with how they'd like to operate in a desired future state.

   - Where does your work come from?
   - Is work coming to you? Or do you have to find it?
   - How many things are you working on at one time?
   - How do you prioritize your work?
   - How do you communicate WIP (work in progress)?

4. Map out how the architecture group will operate (future state) within the current organizational structure and the organizations delivery model. Once completed, make it visible to the rest of the organization.

Play #10

Enable effective decisions

Autonomy is possible only when everyone in the organization knows how their work affects others. Decisions need transparency and collaboration so that they align with the organizational strategy. Simple, universal guidelines can help individuals understand when to escalate decisions.

Understanding if decisions are being made in the right area, at the right level, with clear guidelines on when to escalate will create transparency and enable the organization to shift the decision making to the right area, opening up a space for collaboration and effective decision making to enable the organization to execute on their vision.

Additionally, architecture principles contribute to guiding decision making.

Use this play to:

- Understand how and where decisions are being made today
- Map out what decisions should be made, and at what level
- Take a step towards enabling effective decision making
- Drive decision-making and accountability to the right places in the organization

Expert tip

Empower your organization by shifting decisions down.

Decisions should be made by those closest to the issue / problem to enable effective decision making.

Time required

2 - 3 hours

Who participates

Architects

How often

Every 6 - 12 months

Enable effective decisions

At the end of this play, the architecture group will have transparency around decisions, where they are being made and if/when they are being escalated.

1. **What exists today.** The first step is to capture the decisions being made today. On a physical or digital wall, ask the architects and key domain representatives to individually note down decisions which are being made in the organization. Record the decision, and the business unit (or domain) where the decision is being made (examples below). Give people 5 - 10 minutes to record decisions.

Decision type	Decision	Domain
	Security compliance requirements for third parties	All
	Integration architecture for acquisitions	

2. **Categorise.** Once the group has finished capturing all decisions, talk through each one, ensuring there is a shared understanding. As you do this, start to capture what type of decision it is. Some examples of decision categories include:

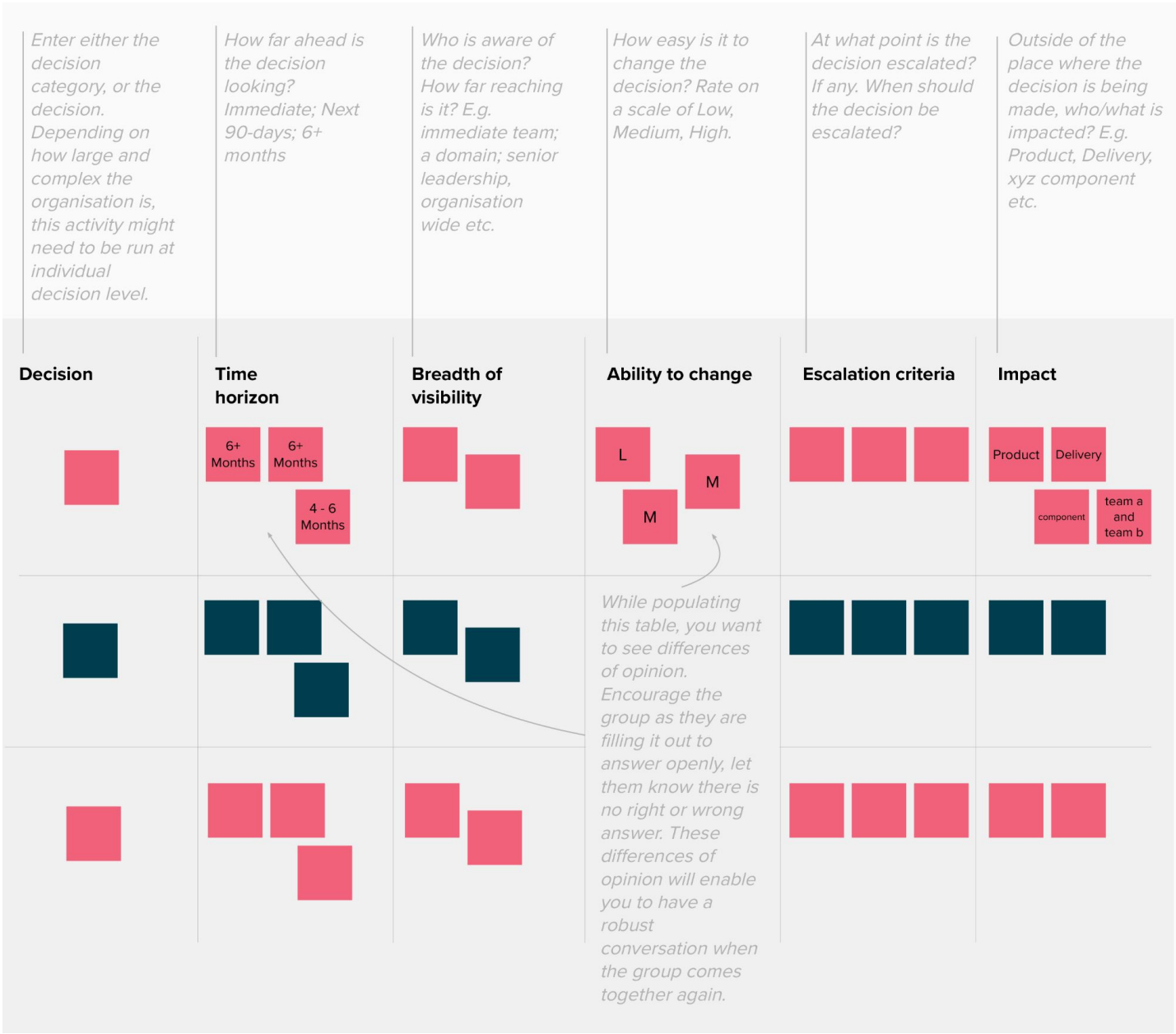
Technology Choices	Operational	System Design	Security	Compliance	Roadmap
Product Strategy	Delivery Prioritization	Tech Platform	Technology Standards	Cross-cutting capabilities	...

3. **Deep Dive.** The third step is to deep dive into the decisions to identify where and at what level decisions are made and where they might need to be further delegated, at what point decisions should be escalated, and how the decisions impact the organization.

   Add columns to your wall/board to capture this information for each of the decision categories. Depending on the size and complexity of the organization, you may need to run this at the individual decision level, rather than the category level.

Example

Enable effective decisions



Play #11

Lightweight architecture decision records

Lightweight Architecture Decision Records (ADR) is a technique for capturing important architectural decisions along with their context and consequences.

Much documentation can be replaced with highly readable code and tests. In a world of evolutionary architecture, however, it's important to record certain design decisions for the benefit of future team members as well as for external oversight.

These details should be stored in source control, instead of a wiki or website, as then they can provide a record that remains in sync with the code itself.

When combined with well-understood Architecture Principles and trust in development teams, this play has the potential to replace much heavier governance processes: because decisions are shared quickly and transparently, it’s possible to address issues after the fact rather than requiring gates and approval meetings.

Use this play to:

- Record decisions for future team members
- Share decisions with external stakeholders
- Avoid burden of traditional processes

Expert tip

You can use automated scripts to detect updates to ADRs in source control and publish a generated report to interested parties.

This could replace much heavier governance or reporting processes.

Time required

15 - 60 minutes

Who participates

Architects, developers

How often

As needed

Lightweight architecture decision records

ADRs should be committed to source control throughout a project.

They should capture **architecturally significant decisions**, defined by Michael Nygard as “those that affect structure, non-functional characteristics, dependencies, interfaces, or construction techniques”.

Each ADR is created in a numbered file (e.g. doc/adr/NNNN-example.md) using a lightweight text formatting language like Markdown. For example:

```
# Title

## Status

What is the status, e.g. proposed, accepted, rejected, deprecated, superseded?

## Context

What is the issue that we're seeing that is motivating this decision or change?

## Decision

What is the change that we're proposing and/or doing?

## Consequences

What becomes easier or more difficult to do because of this change?

## Alternatives considered

What options were evaluated and why were they considered not suitable.
```

For a lightweight toolset to manage ADRs, see Nat Pryce's adr-tools (https://github.com/npryce/adr-tools). It can be used to create and link ADRs in Markdown format. These can be fed to rendering tools like Pandoc for publication in various formats.

Play #12

Artifacts on a page

Decisions are recorded and shared through artifacts. The production of artifacts is a key component to driving a shared understanding of architectural direction and technology strategy.

The right artifacts:

- **radiate knowledge** that enables individuals to align the rest of the business
- **provide transparency** about the decisions being made
- **make decisions traceable** and individuals accountable.

Lightweight Architecture Decision Records is an example of such an artifact.

Use this play to:

- Capture what artifacts exist today in support of the decisions being made
- Highlight gaps and understand what artifacts should be in place to support decisions

Note: Run Play #10 - Enable Effective Decisions before running this play.

Expert tip

When capturing artifacts, focus on those that directly support critical business decisions and have the most significant impact on your organization. Prioritizing artifact creation ensures that your efforts are aligned with strategic goals and that you're providing the most valuable information to stakeholders.

Time required

1 - 2 hours

Who participates

Architects

How often

Every 6 months

Artifacts on a page

At the end of this play, you should have a succinct table (pictured below) that speaks to the core artifacts and from where they are generated. This table will then form part of the **operating model on a page.**

Artefact	Description	As a result of...
Open Design Proposal	A one-page brief for each milestone which summarises the research, spikes and findings which backed the technical decisions for the milestone.	Discovery
Architectural Vision & Principles	Articulation of technology vision and translation into development of an enduring set of directional, fundamental values which guide technology decision-making and activities Each principle comprises a statement, a rationale, and a set of implications. The implications are indicative rather than exhaustive.	Architecture Planning
Architecture Wall	Kanban wall showing architectural work in progress. Important to show work queues and WIP for each person/division. Technology experiments need to be highlighted and also in	Architecture Planning

1. What exists today. After completing Play #10 - Enable Effective Decisions, create a Mural board or Google Sheet to capture the existing artifacts for each decision. Start offline, asking team members to populate the table with their best knowledge. Then, gather everyone for a meeting to review and refine the captured information. Alternatively, if the group is large, conduct a group exercise. This might take 60-120 minutes to capture the current state. The information you should be looking to capture includes:

- Artifact
- Format
- Location
- Owner
- How is it discovered
- How often is it updated
- Does it have version control
- How does collaboration work? How do people suggest amendments
- Issues/concerns

Capture as much information as possible and ensure the group has discussed what exists and why.

Artifacts on a page

2. What should exist. The next step is to run through the list of artifacts and discuss what's missing, needs replacing, or needs an update. Questions you want to run through include:

- Is the artifact the right one for the decision?
- Does the artifact have a clear owner?
- Complete any missing information in the table, e.g. if it doesn't have a clear owner or version control, how can you introduce this?
- Are there any decisions without artifacts?

3. Artifacts on a page. Finally, you should have a clear list of artifacts for the group/organization. Take this list and capture a description for each. The last step is to align the artifact (and decision) with the process from which it originates, producing a succinct table as shown on the previous page.

Play #13

Responsibilities matrix (RACI)

A responsibilities (RACI) matrix clarifies responsibilities and ensures all deliverables and key artifacts have owners to be held responsible or accountable, coupled with those who need to be either consulted or informed.

The RACI matrix brings structure and clarity to describing the roles and responsibilities of a team and their stakeholders and helps teams find alignment on deliverables.

Use this play to:

- Define and document roles and responsibilities
- Provide clarity to the team on who needs to be informed
- Get clear on outcomes and create a shared understanding of who will be responsible and accountable for those outcomes
- Help the team stay focussed
- Provide clarity to the team and their stakeholders on their involvement on delivering outcomes

Expert tip

It's a good idea to go into this with an understanding of who the team's stakeholders are. If you haven't already done so, run Play #6 - Stakeholder Mapping first.

It can be helpful to run Play #10 and Play #12 prior to running a responsibilities exercise too, as there will be decisions to come from those plays which can be fed into the responsibilities matrix.

Time required

60 minutes

Who participates

Architects

How often

As new team members join

Responsibilities matrix (RACI)

RACI Roles

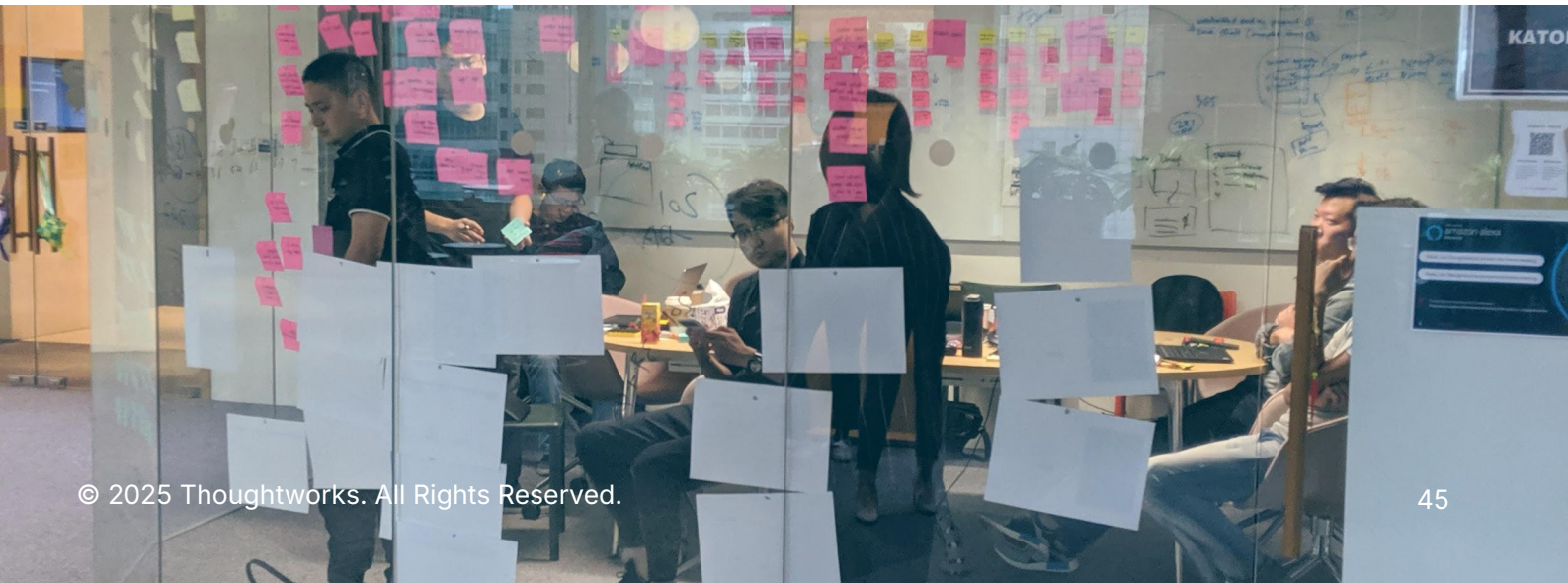
Responsible	The person assigned to complete the task. Several roles (people) can be jointly responsible for a task.
Accountable	The person who takes the final decision and is ultimately answerable for the completion of the task or deliverable. This person usually delegates the work to those responsible.
Consulted	The person who must be consulted before a decision/action is taken. Their opinions are sought and there is generally two-way communication with this person. They are actively engaged and will provide input to the work.
Informed	The person who must be informed about the decision/action taken. They are kept up-to-date on progress, generally there is just one-way communication with this person. They don't contribute directly to the task or deliverable.

Success factors

- You should have only one role marked as Accountable for each task
- You should have at least one role marked as Responsible for each task
- Are there any blanks? Revist and see if you can identify the role
- Share and discuss the matrix with your stakeholders
- Treat this as a living document - review and update as needed

Responsibilities matrix (RACI)

1	**Introduce the RACI roles**	Introduce the four RACI roles to the team (provided below). Discuss the definitions and capture any additional detail for each role relevant to your organization.
2	**Add tasks/ deliverables to the matrix**	Start the matrix by listing out all the tasks/deliverables for the EA function across the top. These will likely be things like the architecture vision and principles, capability maps, decision records and the tech radar to name a few. Not all tasks/deliverables need to be listed, record those most critical to the EA function of your organization.
3	**Add roles/ stakeholders**	Populate the left column with a list of the EA function roles and their stakeholders. Hint, you'll likely be including those on the right side of the stakeholder map. Not every stakeholder needs to be listed, capture those critical to the team.
4	**Fill in the matrix**	Start populating the matrix. Discuss each task and identify who is responsible, accountable, consulted and informed.
5	**Share**	Share the matrix with your stakeholders. It's a good idea to have a group discussion with them to ensure they are comfortable with the roles they've been assigned and there is a shared understanding amongst the group.



Example

Responsibilities matrix (RACI)

	Vision/ Principles	Decision Records	Task 3	Task 4 ...
Enterprise Architect	R	R	I	R
Lead Architect	A	A	R	I
Role 3	I	C	A	A
Role 4	C	I	C	I
Role 5 ...	I	C	R	C

Responsible	Accountable	Consulted	Informed

Play #14

Operating model on a page

An operating model on a page shows how enterprise architecture is executed within the organization and value is delivered to customers.

A simple visualisation will help create a shared understanding. It also helps to highlight any bottlenecks and fragmentations on the architecture group.

It should be a living document and could be a landing page on the wiki. It is recommended to have a pair responsible for keeping it updated and agree on a recurring frequency to look at it.

Use this play to:

- Create a 1-page view of the architecture operating model
- Visualize the architecture operating model within the organization

Expert tip

Ensure the operating model on a page is highly visible within the architecture function and to the organization. It can, and should change over time, revisit every six months and update as required.

Time required

1 - 3 days

Who participates

Architects

How often

Review every 12 months

Operating model on a page

1. Map out the current the operating model. Visualise the organization's operating model and show how Enterprise Architecture sits within it. The visual should depict the value of the Enterprise Architecture function and how the function can help the business meet its objective. Highlight visibility and influence. An abstract example has been provided on the next page.
2. Map out current delivery, technology and architecture processes. You're looking to understand the layers, feedback loops and continuity of work. Get into the detail and capture what feels right for the organization. A high-level example has been provided on the next page. When creating a visual and mapping out the current processes you are looking to understand:
    a. Delivery and product cadences
    b. Releases or specific milestones in place
    c. Feedback loops
    d. Architecture and/or technology cadences
    e. How do architecture and delivery intertwine? Where do they meet? How does work flow between the two?
3. Now you want to bring it all together onto a single page, aim for an A3 size so it can be easily read, printed and shared. There are 4 quadrants to an operating model on a page, more detail is provided on the next page.

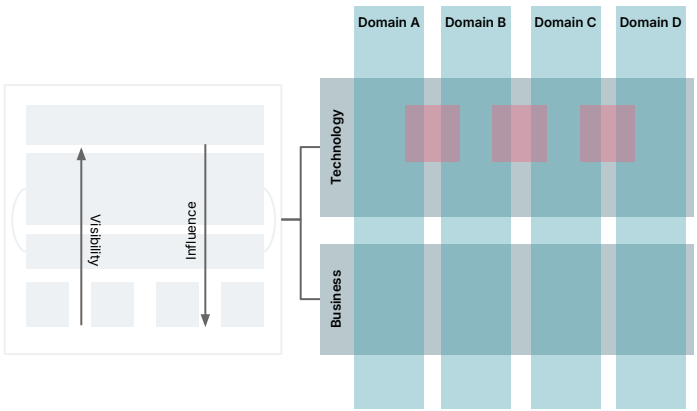
Example

Operating model on a page

Visualize the Enterprise Architecture and organization operating model. The visual should depict the value of the Enterprise Architecture function and how the function can help the business meet its objective. Highlight visibility and influence.

Include artifacts on a page. See Play #12.

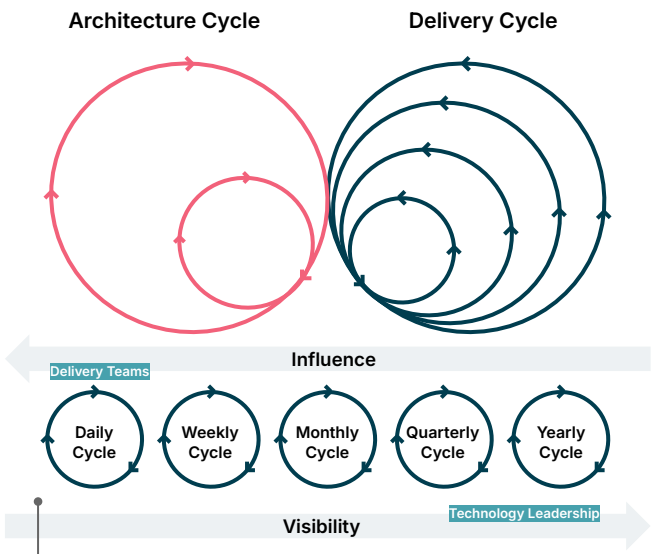
Operating model



Artifacts

Artifact	Description	As a result of
Open Design Proposals	A one-page brief for each milestone which summarizes the research, spikes and findings which backed the technical decisions for the milestone.	Discovery
Architecture Vision & Principles	Articulation of technology vision and translation into...	Architecture planning
Architecture Wall	Kanban wall showing architectural work in progress. Important to show work queues and WIP for each person/division. Technology experiments need to ...	Architecture planning
....	....	....

Processes



Responsibility matrix

	Vision/ Principles	Decision Records	...	...
Enterprise Architect	R	R	I	A
Lead Architects	A	A	R	I
Role 3	C	I	C	I
Role 4...	A	C	C	I

Visualise processes and planning cycles. Show how architecture fits with delivery.

Include a responsibilities matrix (RACI). See Play #13.

Play #15

Capability mapping

Capability mapping will help you answer two questions:

1. What do we have to be able to do?
2. How good are we at it?

Capabilities describe an organization's operating model in terms of goals and competencies (what is to be done), rather than implementation specifics (how things are done).

Capability models provide a description of the business context that is stable enough to serve as a basis for identifying and prioritising technology and process initiatives over a medium to long term. Because capabilities describe the what and disregard how things are done and by whom things are done, the map (and what an organization does) remains fairly stable.

Use this play to:

- Create a holistic view of the organization and **what** it does, using a common language
- Get clear on where the organization stands
- Create a base from which you can start mapping out a desired model (future state)

Expert tip

Organizations can be stuck in a pattern of using technology assets and product names to describe what it is they do.

Take your stakeholders on the capability mapping journey with you to ensure you have their buy-in and you're creating a shared understanding.

Time required

1 - 2 weeks

Who participates

Architects, business, technology, product stakeholders

How often

Initial mapping then review every 6 - 12 months or when key strategic decisions are being made

Capability mapping

Customers

Customers are the people or services who consume the capability. They could be internal or external.

Channels

Channels, such as mobile apps, website, stores, social media etc., describe how your customers engage with your experiences.

Experience capabilities

Experience capabilities describe the experiences an organization offers to their customers, both internal and external customers by way of products and features.

Business capabilities

Business capabilities describe an organization's operating model in terms of what they do. They describe what is to be done, rather than implementation specifics - how things are done.

Each business capability encapsulates the processes, data, systems, events, and people that provide that capability.

Technology capabilities

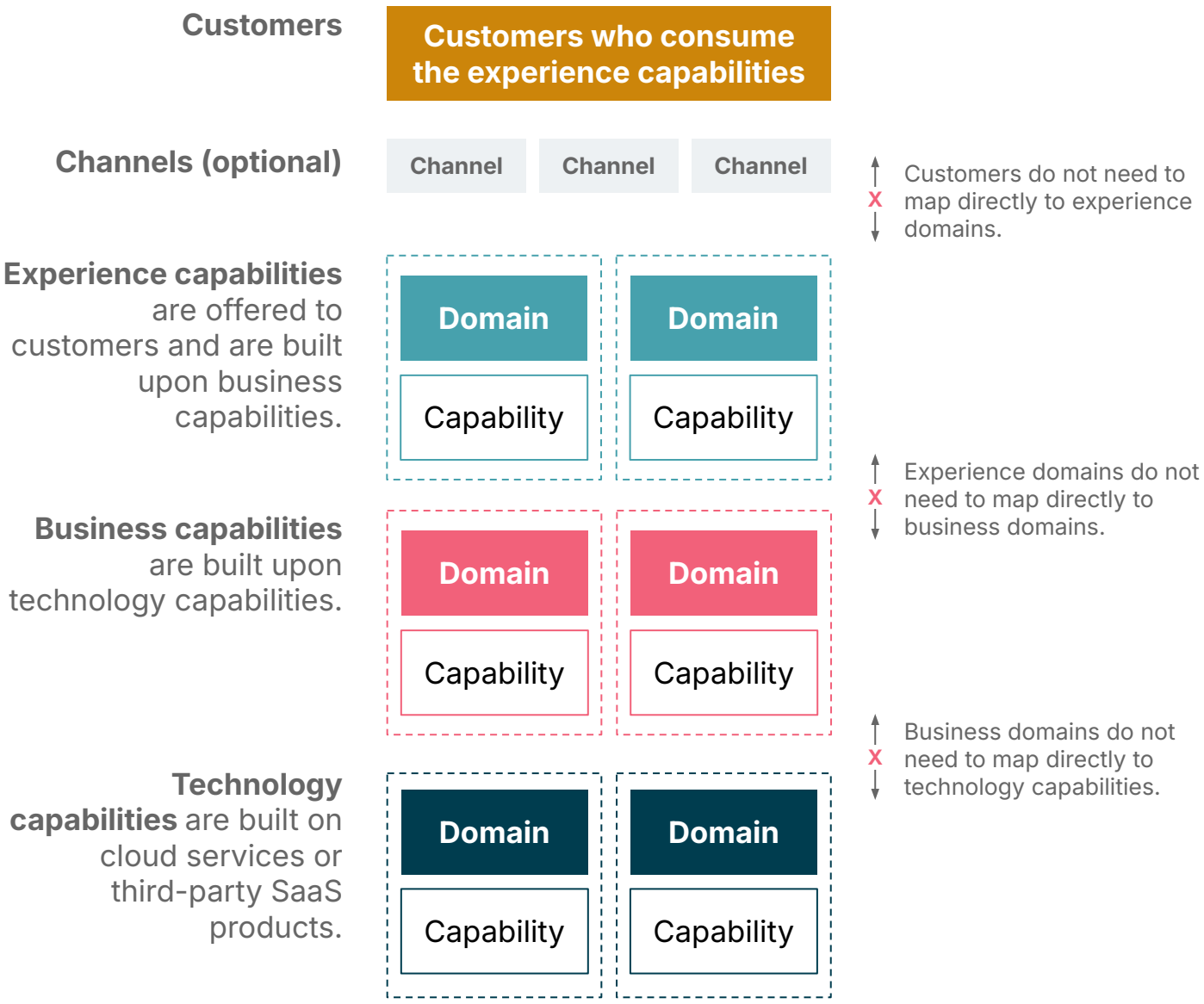
Technology capabilities are the fundamental building blocks that enable the organization to build, deploy, monitor and operate applications.

Capability mapping

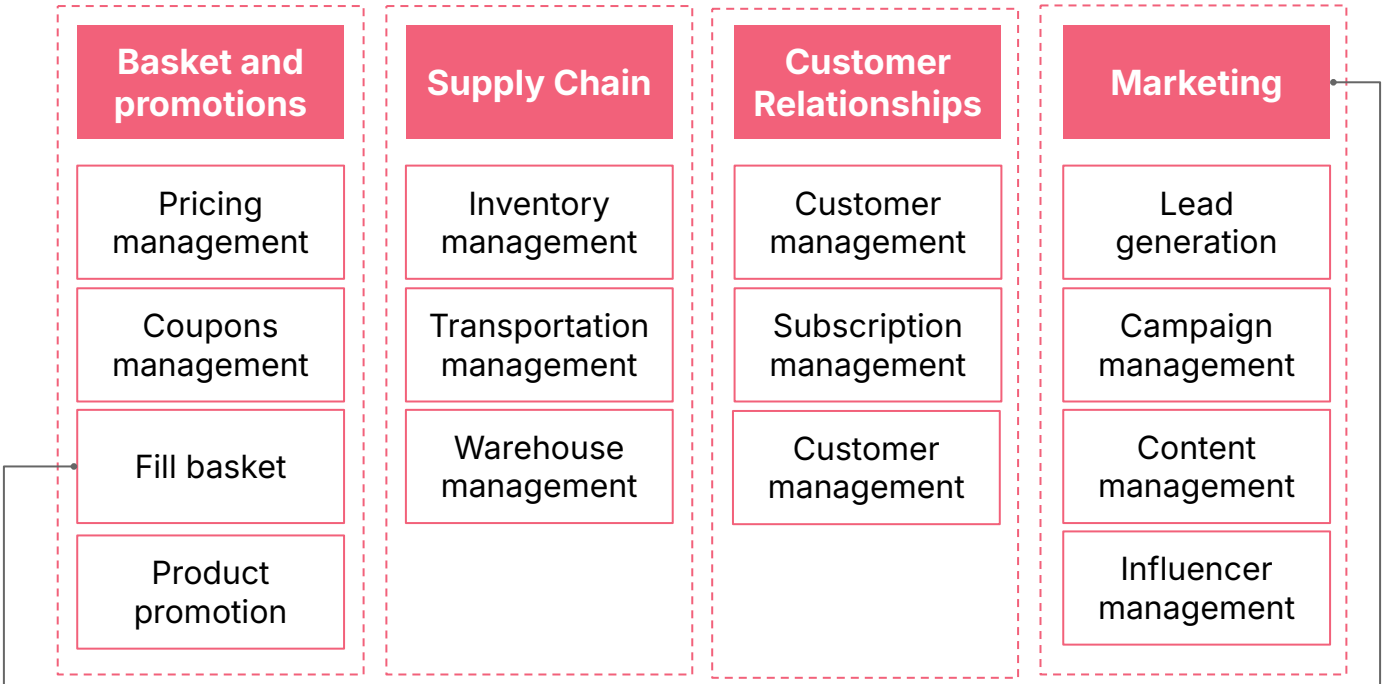
Capabilities are building blocks of functionality, expressed not in terms of systems or teams, but in **what they enable someone to do**.

Capabilities represent what the organization needs to be able to do to execute its strategy (Lynch et al., 2003) - capabilities may be implemented through building, buying or reusing existing offerings within the organization.

A capability map consists of 4 layers, starting at the top with the customer.



Capability mapping



Strong Capability Ownership
The technology supporting any **business capability** must have a clear technical and business owner. A capability is a product offered to the rest of the organization and the product owner should be accountable for creating a compelling offering. Capabilities may be grouped into clusters known as “domains” where each domain is a black box with clearly accountable ownership.

Independent Domains
To manage the complexity of technology in a large and diverse business, boundaries must be drawn around clusters of related capabilities and technology solutions confined within those boundaries. These clusters of capability, **domains**, should form a natural complexity boundary around the capability clusters.

For large, complex organizations there may be a need to introduce **subdomains** to support more specific use cases. A subdomain is a smaller cluster of capability within a domain.

Capability mapping

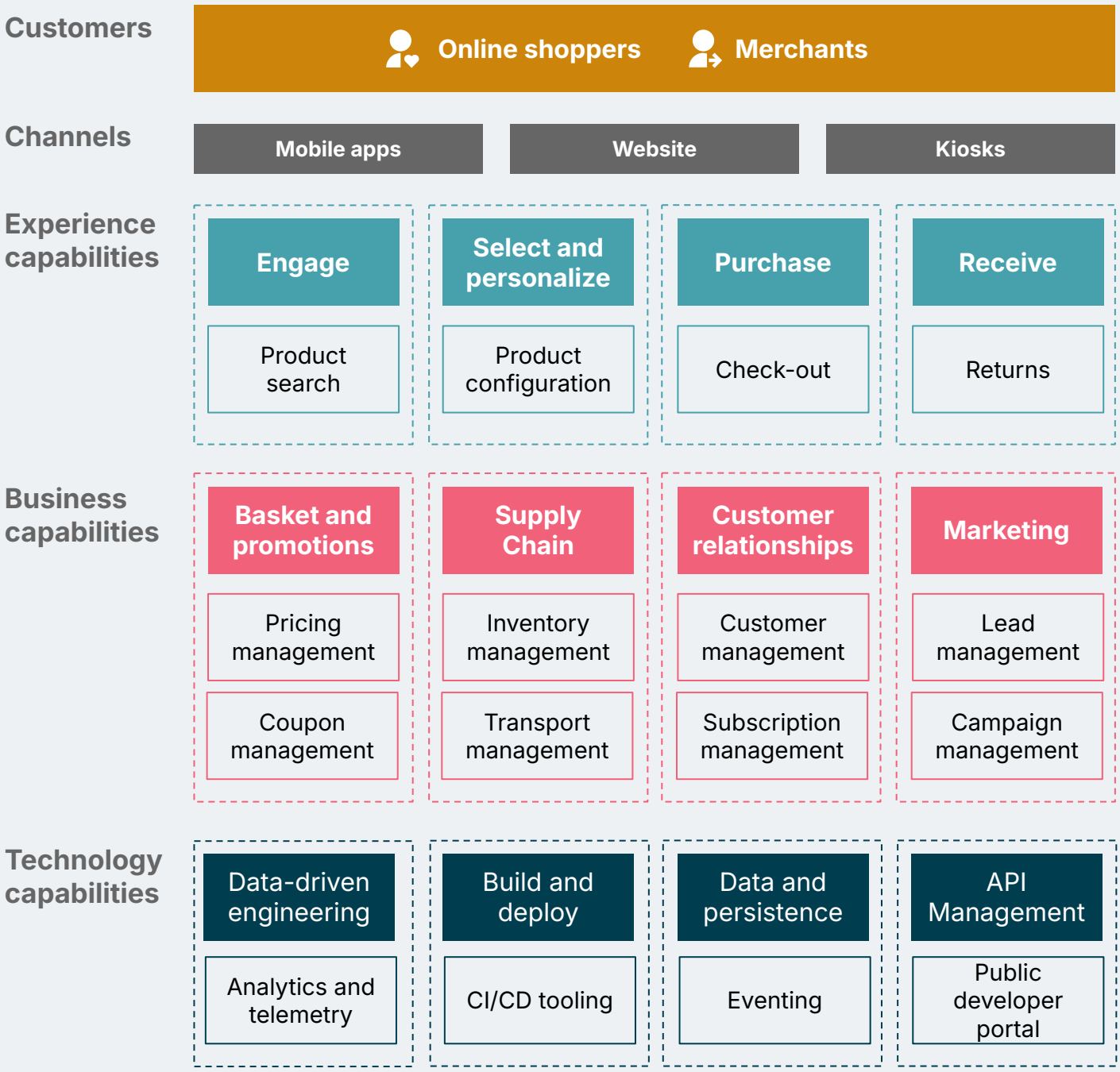
1	**Identify customers**	Customers sit at the top of a capability map. Identify and capture all of the organization's customers
2	**Capture capabilities**	Through desk research, interviews and workshops, start to create a view of the organization's capabilities. Start by focussing on business capabilities, but don't stop yourself from capturing product and technology capabilities if they come up as you go. The key is to get capabilities on the page. They are not going to be right the first time, but you have to start somewhere. Don't worry too much at this stage what domain they belong to or exactly where they sit in terms of business/product/technology. Just get a mass of capabilities on the page.
3	**Refine and group capabilities**	Take your ~~mess~~ mass of capabilities and start to organize into layers (technology, business, product). Create buckets of related capabilities. Review the terminology used to describe the capabilities and refine as you go. You may still be using product names or technology asset names - try to abstract what the product/asset enables someone to do.
4	**Validate and seek feedback**	If you aren't doing this hand-in-hand with key business, product and technology representatives, share back this early draft for validation and feedback.
5	**Define domains**	Take your capability buckets and define domain names for each. You might find one bucket becomes 2 domains, or 2 buckets are merged together as a single domain. These domains probably don't exist in the organization's current state and that is okay. When defining domains, more often than not you are referring to the future state.

Capability mapping

6	**Review and reflect**	Do your capabilities describe what they enable someone to do? Do product and technology asset names still feature in the map? Go back over the domains and capabilities and check they describe what they enable someone to do.
7	**Share and validate**	Share for feedback and validation with key business, product and technology representatives. Take feedback with a grain of salt. Key business, product and technology representatives may not always be right. Attachment to a particular product or service or team, and fear of what might happen to that product/service/team may be driving the feedback.
8	**Refine and align**	Do one final pass of the capability map. Make sure domain blocks and capability blocks are the same size and spaced evenly. Check domains make sense. Review the language used in capabilities. Add in definitions of each capability.
9	**Publish and share**	Once you feel the map is in a good state, publish it in an accessible location and share it with key stakeholders. Encourage them to share it with their peers and teams. Consider running a session or two to take people through the map, clarify key concepts and what is comes next.
10	**Add depth**	Once you have the base capability map you can add as much depth to the model as desired. A capability model can be used to remodel technology services and teams. Layers can be added to highlight duplicate capabilities, capabilities provided by third-parties, capabilities that are used by multiple business units/capabilities on the customer critical path. The options are endless, it all comes down to the story you want to tell. The following plays run through how to approach some of these options.

Example

Capability mapping



Conway's Law asserts that “organizations are constrained to produce application designs which are copies of their communication structures[1].” This often leads to unintended friction points.
The 'Inverse Conway Manoeuvre' recommends evolving your team and organizational structure to promote your desired architecture. Ideally your technology architecture will display isomorphism with your business architecture.

Play #16

Capability assessment

Increasingly, organizations are under pressure to do more with less. Capabilities must be built fast and be adaptable to changing conditions.

A capability assessment provides clarity on which capabilities create competitive advantage over those of business necessity. It enables the organization to be explicit in what capabilities are core to executing the strategy.

Not all capabilities need to, nor should be, custom built. A capability assessment will highlight capabilities that are commoditised in the market and can be bought or outsourced to free up talent within the organization to focus on building capabilities that provide competitive advantage.

Use this play to:

- Understand which capabilities create competitive advantage for the organization, which provide strategic support and those that are business necessity
- Assess which capabilities the organization should build, buy or partner on

Expert tip

Before undertaking a capability assessment, the organization's purpose, vision and strategy must be publicised and accessible, and Play #15 - Capability Mapping complete.

An alternate to this play is Wardley Mapping.

Time required

4 - 6 hours

Who participates

Architects, key business, product, technology representatives and strategists

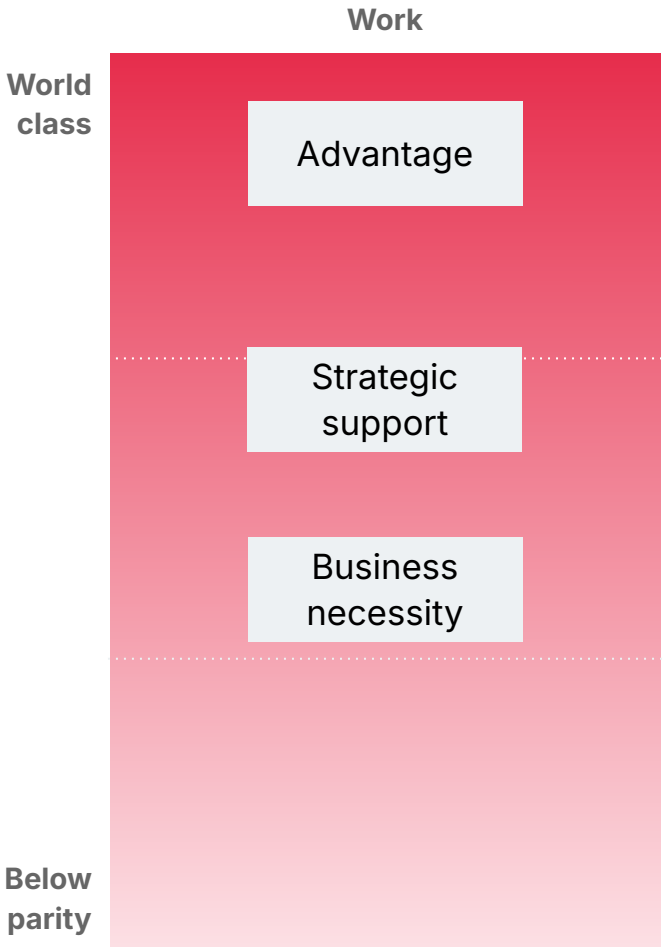
How often

Every 6 months or as needed

Capability assessment

Types of work helps to determine competitive advantage and how to set goals as described in The Capable Company by James F. Dowling, John G. Diezemann, and Richard L. Lynch. Lynch et al., 2003 state that being clear about the different types of work helps a business allocate the right level of time and resources toward building the appropriate capabilities in these areas.

The table below is taken from The Capable Company and describes types of work.



- Needs to be performed at a world-class level
- Needs to be built fast
- Measured across industries
- Creates competitive advantage - differentiates the organization's offering

- Needs to be performed above industry parity
- Measured within industry
- Leverages the advantage capabilities

- Needs to be performed at industry parity
- Designed for efficiency - low cost

Capability assessment

1. Start with a shared understanding of the organization's purpose, vision and strategy. Before assessing capabilities you, and those involved in the activity, need to have a shared understanding of the organization's purpose, vision and strategy. Meet as a group to discuss - even better, have someone who was involved in creating the purpose/vision/strategy come along to walk through it and answer questions. Don't assume there is a shared understanding.

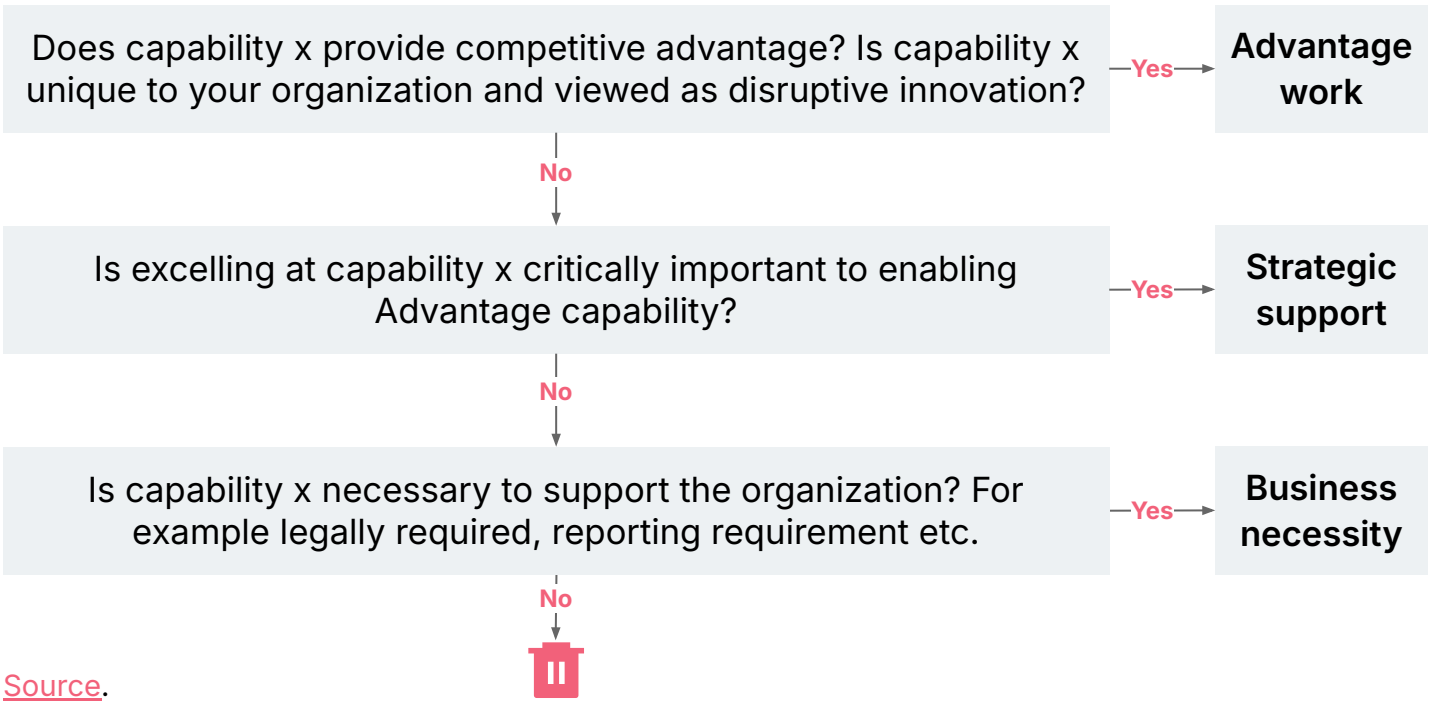
Purpose: Reason to exist

Vision: Aspiration for the future

Strategy: Alignment of opportunities, plans & behaviour

2. Assess what type of work each capability is. With a solid understanding of the organization's purpose, vision and strategy, assess each capability in terms of what type of work it is (advantage, strategic support, business necessity).

You could do this as a (lengthy) group discussion, or ask participants to do it offline (via a survey or digital board), collate opinions, then review as a group and finalize.



Source.

Capability assessment

3. Assess what sourcing option to take for each capability. For capabilities that fall into the advantage work bucket, they must be built internally as a compelling product to be offered both internally and externally. For all other capabilities, map them against the below table which has been adapted from The Capable Company.

	Business necessity work	Strategic support work
Demands unique company knowledge	Manage for efficiency	Build
Does not demand unique company knowledge	Buy/Outsource	Partner

Build: The capability is a differentiator and provides competitive advantage. Build and manage this capability internally.

Buy/Outsource: The capability is not of strategic advantage and is commoditized in the market. Look to acquire/buy or outsource the capability.

Partner: Look to partner with another organization to deliver the capability.

Manage for efficiency: Look to produce at the shortest cycle-time and lowest cost. Extra investment in these capabilities won't produce greater returns.

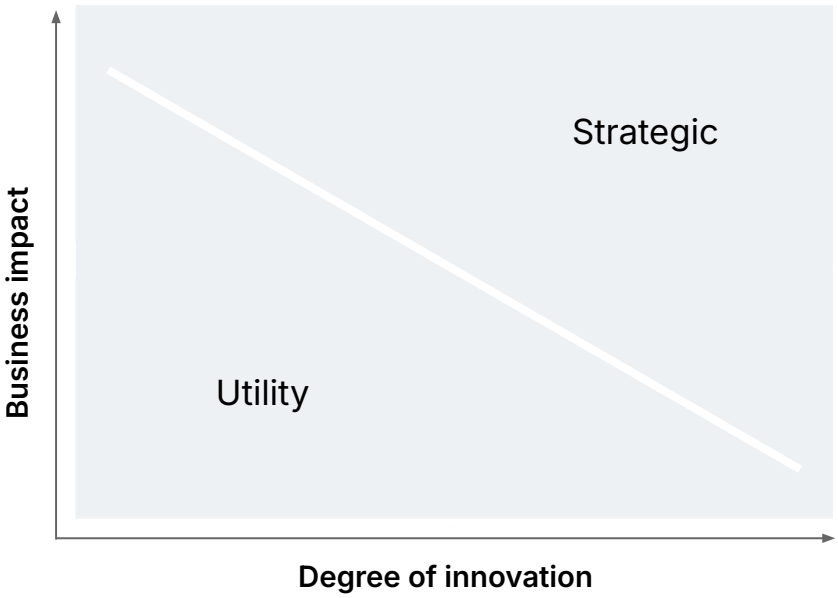
4. Reflect on the capabilities. Are there any capabilities missing that are needed to execute the organization strategy and realize the vision?

Capability assessment

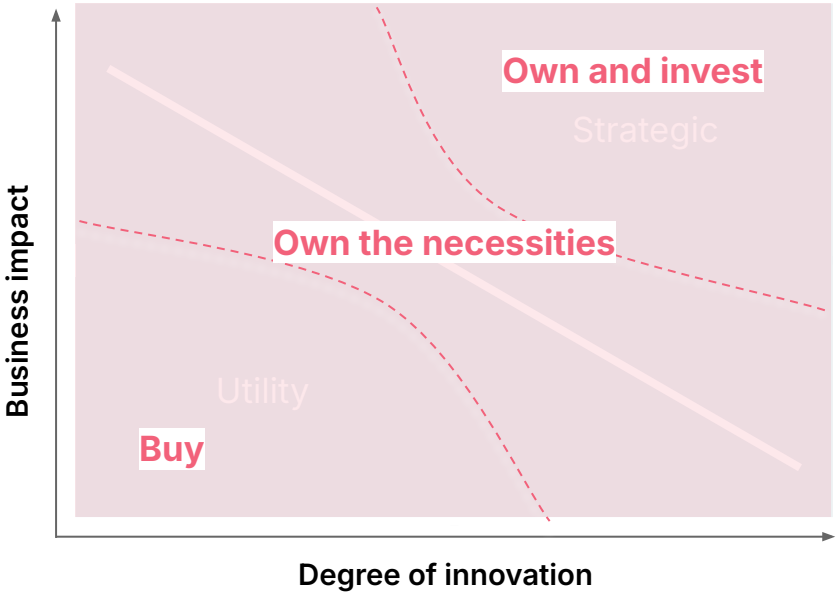
5. Further assess capabilities against dimensions relevant to the organization.

This is another useful model when honing in on differentiation and applying a deeper line of thought on the desired level of innovation for each capability.

Start with the base model (pictured right); map the capabilities against business impact and degree of innovation.



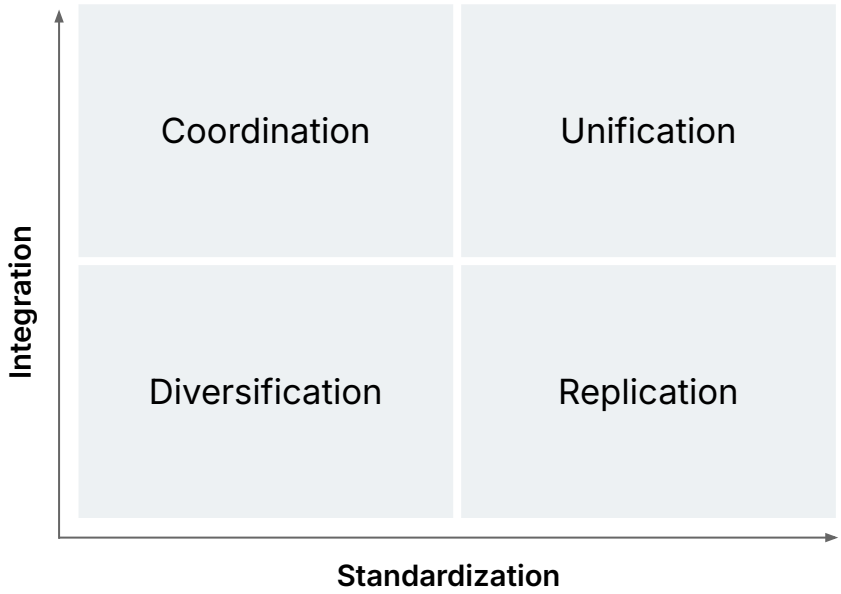
Next, apply the following overlay to ascertain which capabilities are the differentiators (own and invest) over those that are commoditized (buy).

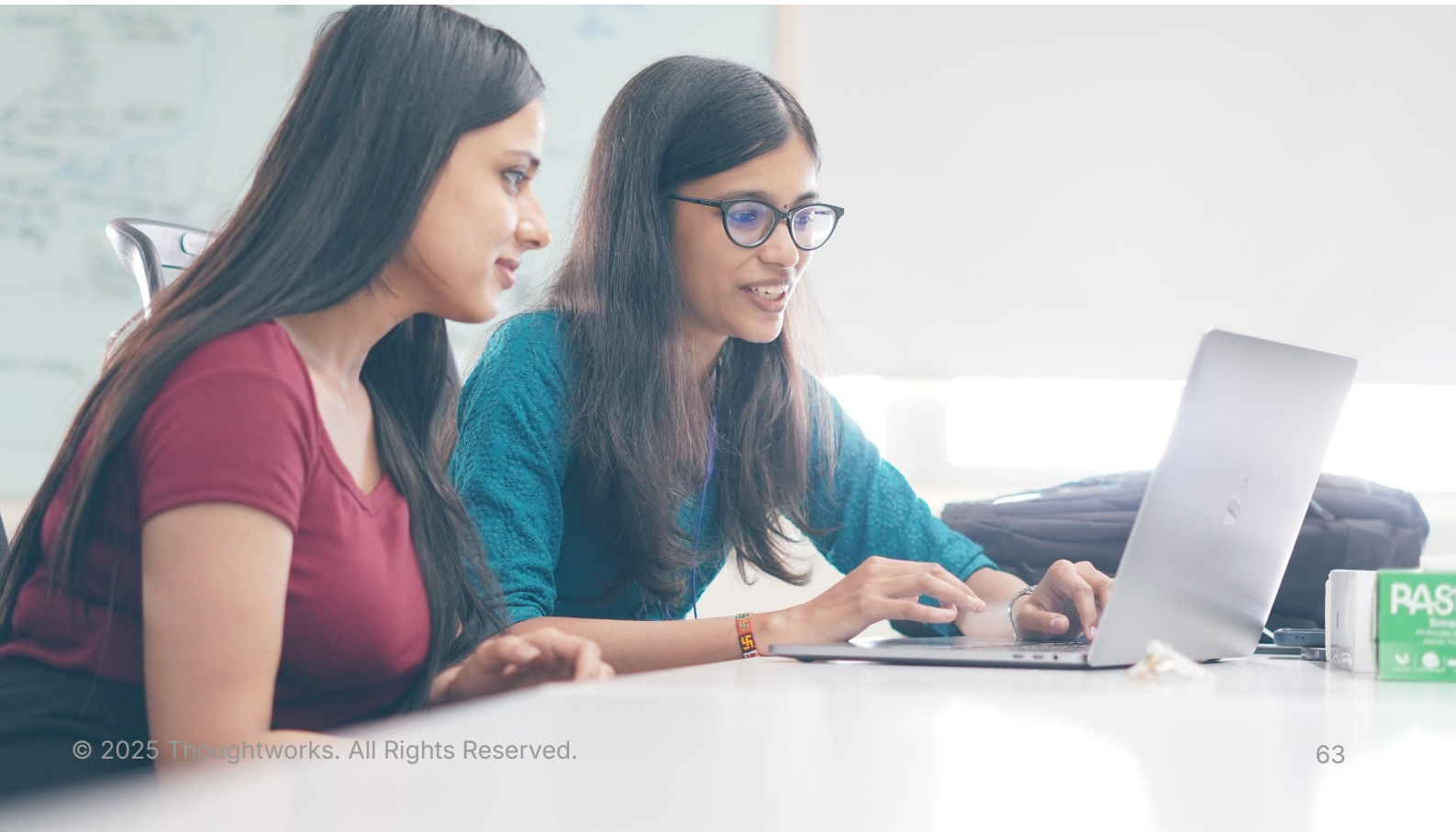


Capability assessment

5. Further assess capabilities against dimensions relevant to the organization (continued)...

This Operating Model Quadrant from Enterprise Architecture as Strategy, Ross et al, 2006, is relevant for organizations wanting to unify capability across their group of companies, or when addressing consolidation of inorganic growth.





Play #17

Capability heat map

A heat map is a tool to visualize a range of different aspects. Capability heat maps visually tell a story using common language. They are useful for analyzing and communicating the current state of the organization's capabilities and to where attention should be directed.

Heat maps can be used to portray capabilities from various perspectives, including:

- Cost
- Revenue
- Performance
- Repeatability, adaptability, scalability
- Maturity

Use this play to:

- Visually represent the current state of capabilities within the organization
- Easily communicate a story - the data can be understood at a glance
- Analyze capabilities through different lenses
- Identify areas for improvement
- Identify areas for investment

Note: Run Play #15 - Capability Mapping, and Play #16 - Capability Assessment before getting started.

Expert tip

When creating a capability heat map, the key to success is collaboration and transparency. Involve a diverse group of business, technology, and product stakeholders from the start, ensuring that the insights and actions are collectively owned.

Time required

2 - 3 hours

Who participates

Architects, business stakeholders and SMEs, developers, technology leaders, product folk

How often

As needed or every 6 months

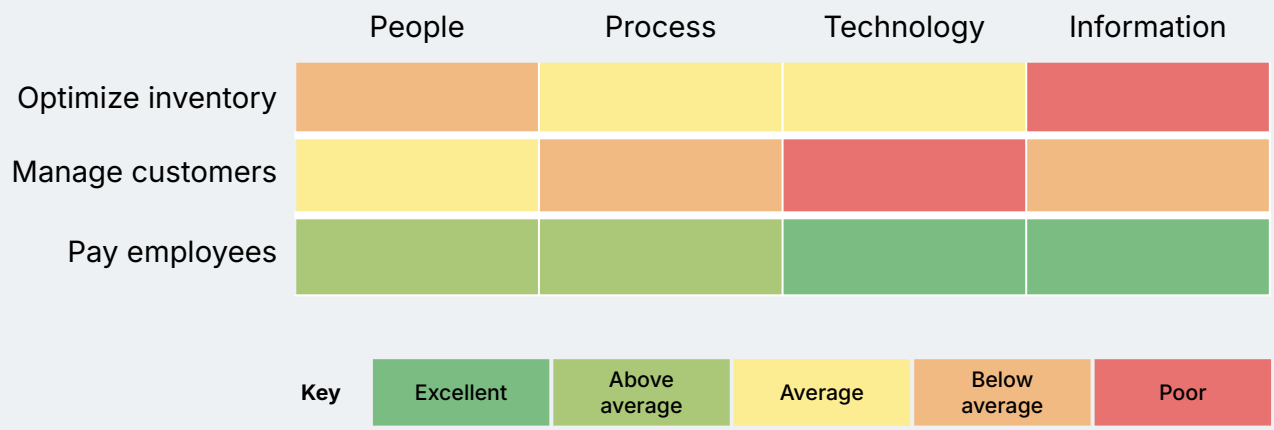
Capability heat map

1	**Identify aspects to be assessed**	Start by identifying aspects to be assessed. You might be assessing more than one aspect, figure out if the aspects should be represented on a single map or multiple maps.
2	**Assess the capabilities**	Across two, 90 minute workshops, assess the capabilities as a group. This can be done as a group of architects, with the outputs being presented back to a wider group of stakeholders and interested parties. Or, alongside key business, technology and product representatives. As you go, capture brief notes on why a capability has been graded a certain way. Attach these notes as an appendix when you publish the heat map.
3	**Playback the heat map**	If you've created the heat map/s siloed from key business, technology and product representatives, play it back to them for feedback and input.
4	**Determine actions to address hot spots**	Together with key business, technology and product representatives, determine actions to address hot spots. Doing this with key representatives from across the organization ensures you have their buy-in on initiatives/work to be done. Coming to them with a list of actions will usually get push back. Create the actions collaboratively so they have a sense of ownership and control.
5	**Publish the heat map**	Publish the heat map so that everyone has access to it. You want it to be visible so that a) you're being transparent and making information accessible, and b) so that everyone knows the criticality of initiatives and the why behind their work.

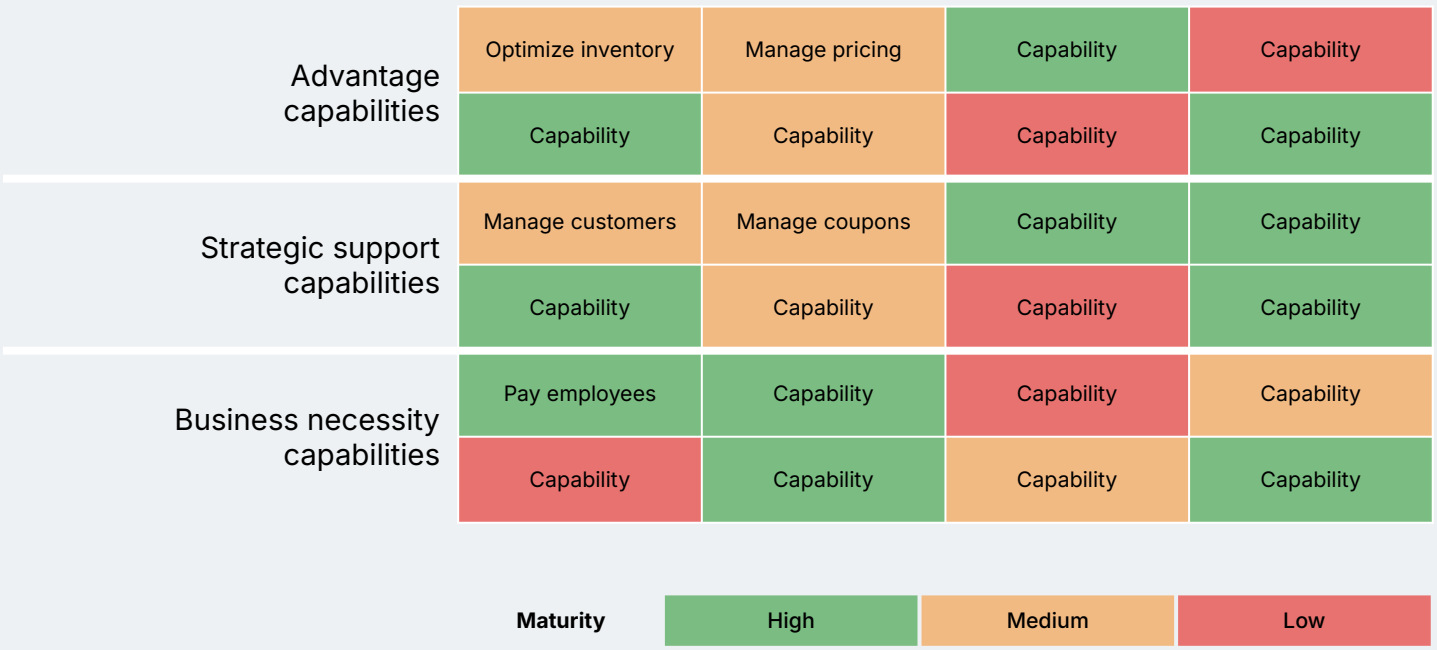
Example

Capability heat map

Capability performance heat map



Capability maturity heat map



Play #18

Capability gap analysis

Gap analysis looks at where the organization is today, and compares that to where the organization needs to be to be competitive in the market and realize their vision.

Gap analysis is a quick way to understand capability gaps in the organization and develop a plan and approach to address the gaps in a streamlined way.

Capability gap analysis takes a holistic view of the organization, enabling you to make informed decisions.

Use this play to:

- Understand the current state
- Understand the required or ideal future state
- Assess the gap between current and future state
- Understand where to focus the organization's time, money and people

Expert tip

To assess capability gaps, you will want to gather different opinions. Someone who is close to a particular capability might bring bias into the assessment. Therefore, it is good to sense check with a number of people where possible.

Time required

1 - 2 days

Who participates

Architects, key business, product, technology representatives and strategists

How often

Quarterly

Capability gap analysis

1. Assess current state vs required state. Together with key business, product and technology representatives, workshop each capability against the table below.

To save time and have a more productive workshop, it might be beneficial to have folks complete the gap analysis via a survey first, collate the results, then bring the collated view to the group to discuss and confirm.

Capability	**Current state** *Level 1 - 5*	**Required state** *Level 1 - 5*	**Gap** *Required state less current state*
Manage pricing	3	5	2
Manage coupons	2	4	2
Manage customers	2	5	3
Manage subscriptions	4	4	0
Optimize inventory	1	5	4

1	**No capability**	The capability does not exist in the organization
2	**Some capability**	Little knowledge of the capability. Likely unstable.
3	**Developing capability**	Requires effort to manage, maintain and execute
4	**Developed capability**	The capability is mature, stable and performant
5	**Optimized capability**	Repeatable, stable, scalable and adaptable capability

Capability gap analysis

2. Prioritize capabilities. With your list of gaps, and output from the capability assessment, use the framework below to prioritize the work to reduce the gaps.

This prioritization framework has been taken from The Capable Company: Building the Capabilities that Make Strategy Work.

3. Identify actions to close the gaps. Starting with those of highest priority, identify what needs to be done to close the gap.

Capability	**Gap** *= required state - current state*	**Type of work** *Advantage, strategic support, business necessity*	**Prioritization** *High, medium, low*
Manage pricing			
Manage coupons			
Optimize inventory			

Prioritization key

Gap	Business necessity	Strategic support	Advantage
> 2.5	Medium	High	High
1.5 - 2.5	Low	Medium	High
<1.5	X	Low	Medium

Play #19

Wardley mapping

Wardley Mapping is a strategy framework and communication tool. The Wardley Mapping technique increases awareness of the organization's strategic situation through a shared visual map. It enables you to gain a deep understanding of your user's needs and what the organization needs to do to build value for the user and operate efficiently.

Wardley maps combine value chains and evolution, enabling you to deeply know the organization and to make good strategic decisions.

The Wardley Mapping technique can be used to assess components. A component can be a capability (and the underlying technologies and practices that bring the capability to life), along with knowledge and data. From here, components will be used to describe all of these elements (capabilities, technologies, practices, knowledge and data).

Assessing components with the Wardley Mapping technique is done whilst taking into consideration the user, their needs, the value the component delivers to the user as well as to the organization.

Use this play to:

- Assess components in relation to the user, their needs and the value the component delivers
- Assess capabilities in respect to their own market
- Strategically assess build, buy or partner options
- Create a simplified view of the organization

Expert tip

Maps are not still, but constantly changing. Mapping something like this is a collaborative activity and is not meant to be a done-once then show-it-to-all approach. The mere exercise of collaboratively mapping this strategy will help quickly align various stakeholders.

Time required

1 day - 3 weeks

Who participates

Architects, key domain stakeholders

How often

At least yearly

Note: this play has been derived from Simon Wardley's Wardley Maps

Wardley mapping

Start by creating a value chain. Value chains are a simplified model of an organization. You probably can't capture the entire organization within a single value chain, create individual value chains for domains, specific services etc.

A value chain needs to be anchored to user needs. You need to understand who your users are before you start (you will create different value chains for each user type). If you don't have a clear idea of your users and their needs, first, create a user journey map for all users so you know who they are and what their needs are.

User types

Customer - Pays for the product. For example, advertisers on Facebook.

User - Uses the product. For example, people who connect with each other on Facebook.

Service receivers - Non-paying customers who don't use the product, but need to be served. For example, shareholders, internal users (employees), regulators, tax office, ombudsman and external users who use the product because it is in someone else's interest.

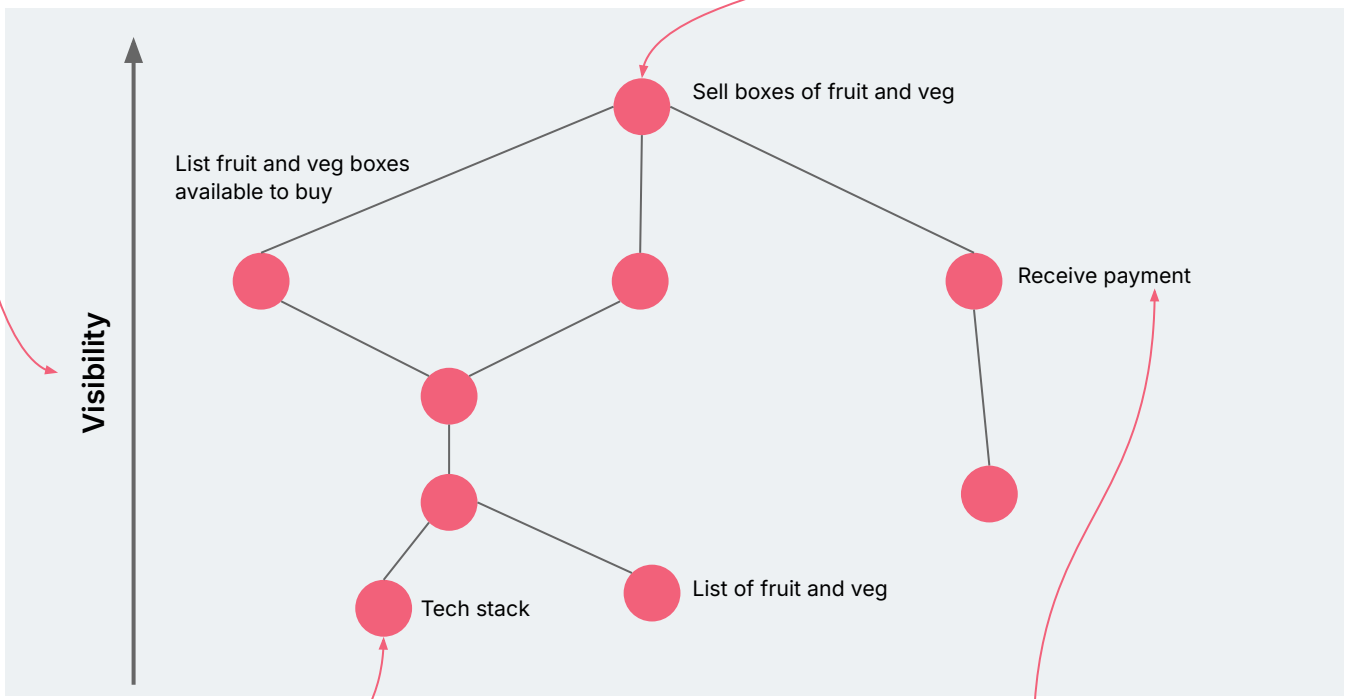
Example value chain

Wardley mapping

Create a value chain.

At the very top of the chain is the component you care most about. For example, if you were an online fruit and veg retailer; selling boxes of fruit and veg would be the thing you care most about and would therefore sit at the top of the value chain.

The higher up the chain components are, the closer they are to the user and the greater visibility the user has of the capability.



Once you've identified what sits at the top of the value chain you want to ask, what you need to deliver that? In the example provided here you need to list your stock that is available to buy and you need to receive payment. Keep going. For each component you add, ask what does that component need to deliver value?

You can aggregate components and store the details of the aggregated components in another map. If not, the value chain will become very large and complex.

Wardley mapping

Once you've created your value chain/s, you want to assess each component in respect to its own market. Move the components left and right so they sit in the right spot on the evolution axis.

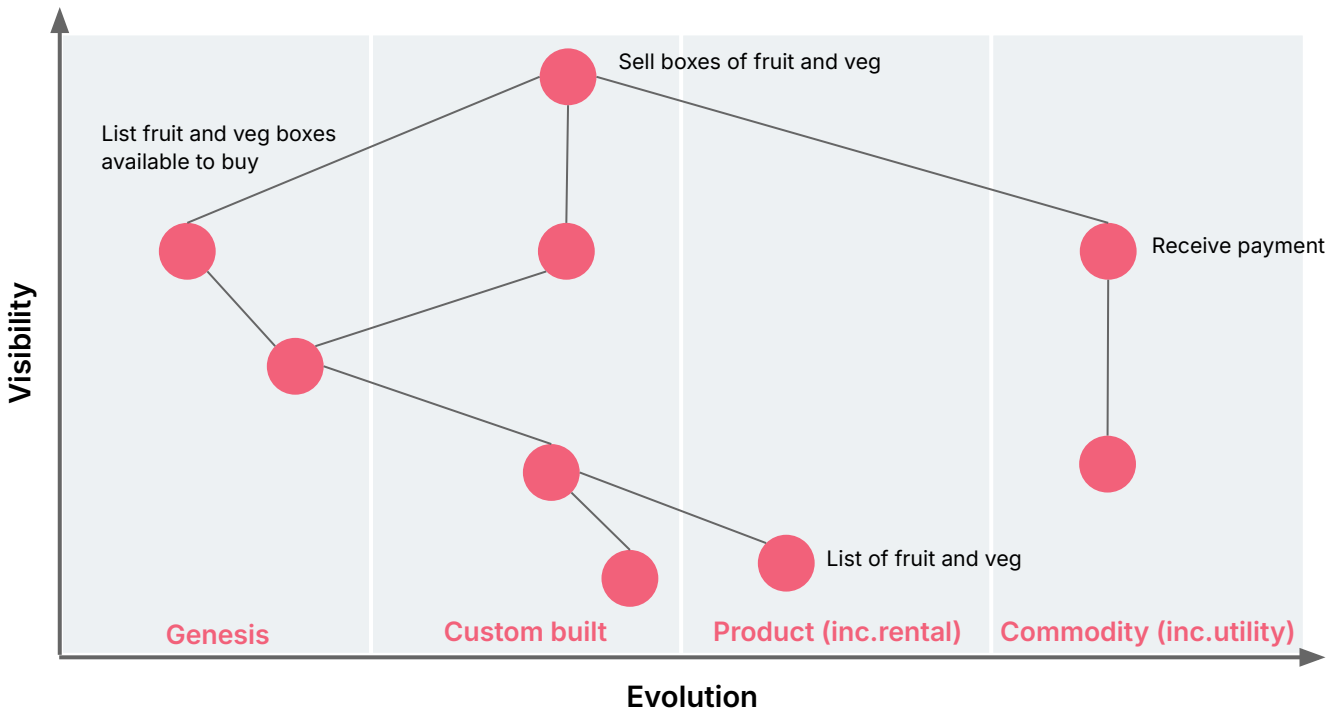
Evolution axis

Genesis - (novel) Immature, unstable, unique, constantly changing, newly discovered.

Custom built - (emerging) Something that is individually made and tailored for a specific environment.

Product - (good) The increasingly common, the manufactured through a repeatable process, the more defined, the better understood. Change becomes slower here.

Commodity - (best) This represents scale and volume operations of production, the highly standardised, the defined, the fixed.



Wardley mapping

Consider the approach required to develop the components. There is no one size fits all when it comes to the approach to develop components, as they move through evolution the approach will change.

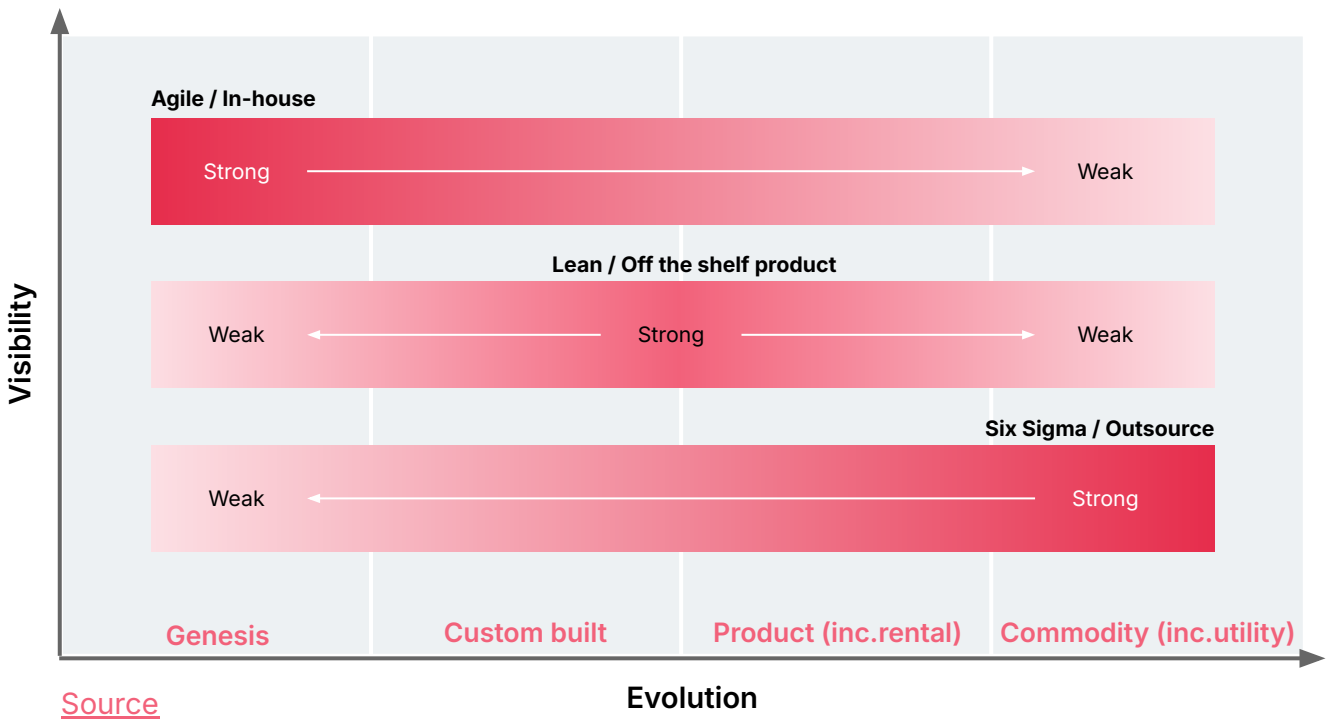
Approach

Agile - excellent for developing components in the genesis phase where experimentation is needed. Agile helps removes uncertainty and adapts to change.

Lean - excellent for nurturing components in custom built and product phases where the focus shifts to improving. Lean helps to refine value and reduce waste.

Six Sigma - excellent for managing components in the commodity phase where scale and stability are key. Six Sigma focuses on reducing process variation and increasing performance.

Outsource - consider components in the commodity phase for outsourcing. If they exist in the market what will you gain from recreating them in-house?

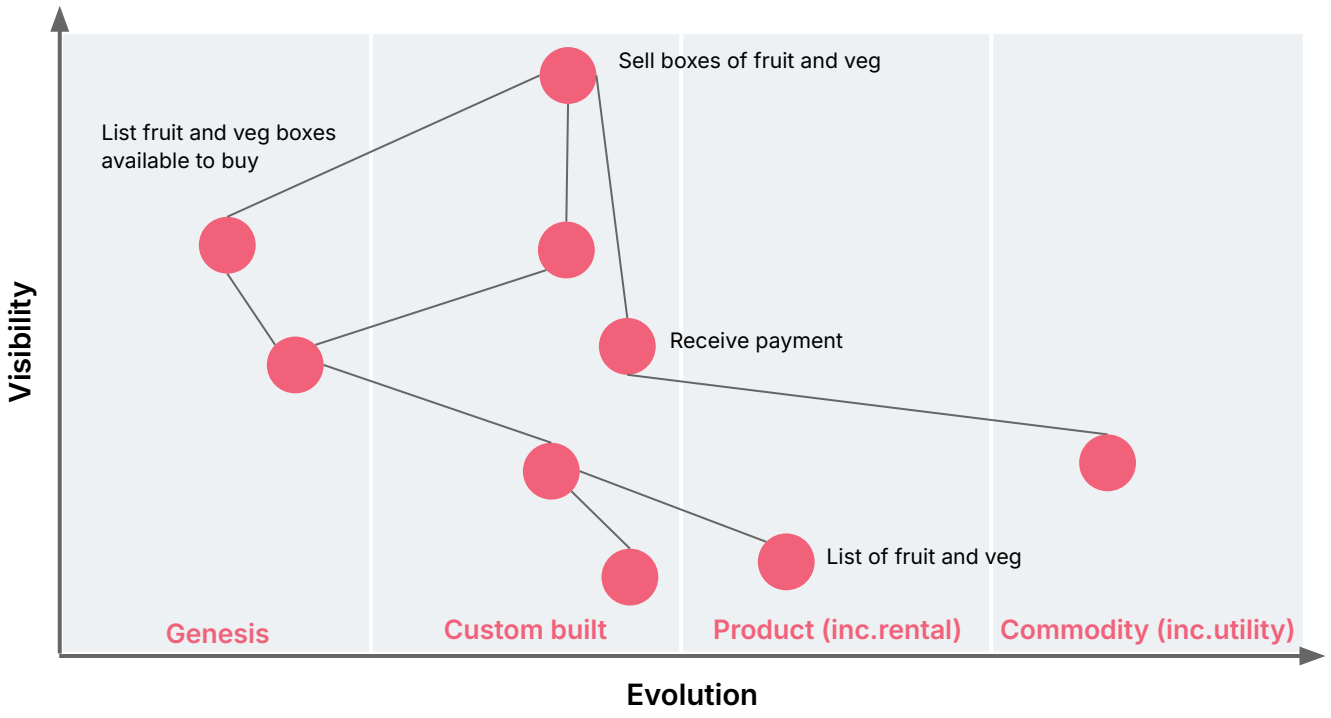


Source

Wardley mapping

How to read the map and assess your business.

- A healthy map has distribution on the left and right side of the evolution axis
- The left hand side of the map is the uncharted zone - components there are new, ever changing and have no associated know how
- The right hand side of the map is the stable zone - components here are mature, reliable and rarely fail
- Introducing changes to mature components is extremely difficult
- Building a service on top of immature components is a challenge
- What has been custom built that is a commodity in the market? For example, receive payment doesn't need to be custom built, there are many solutions that exist today, stop building and maintaining this and outsource it
- Is there something you have built, that is very stable, sits on the right hand side of the map and doesn't exist in the market? You might have just found a new business opportunity
- You want to be building your business on stable components
- Wardley Mapping can be a nice feedback loop to the business strategy - often opportunities for competitive advantage and disruption can come about [Source](#).



Play #21

Architecture roadmap planning

A roadmap can be a great strategic communication tool for guidance and direction under the following considerations:

- It's always a work-in-progress
- It's a collaboration tool and needs to bring everyone's views into alignment (even if that means working through disagreement)
- The conversation about it is almost more important than the picture
- It needs to be flexible and adapt with the organizational change of priorities

A roadmap should be written in terms of business capabilities and should help others see the vision of what you're doing.

Use this play to:

- Create a strategic communication tool
- Establish expectations and set direction
- Steer your technology organization toward delivering on the business goals

Expert tip

A roadmap is a living artifact - you must commit to reviewing it each quarter. Does the roadmap still make sense? Does it need reprioritizing? Gather your stakeholders and have a conversation around it.

Time required

1 - 5 days

Who participates

Architects

How often

Quarterly

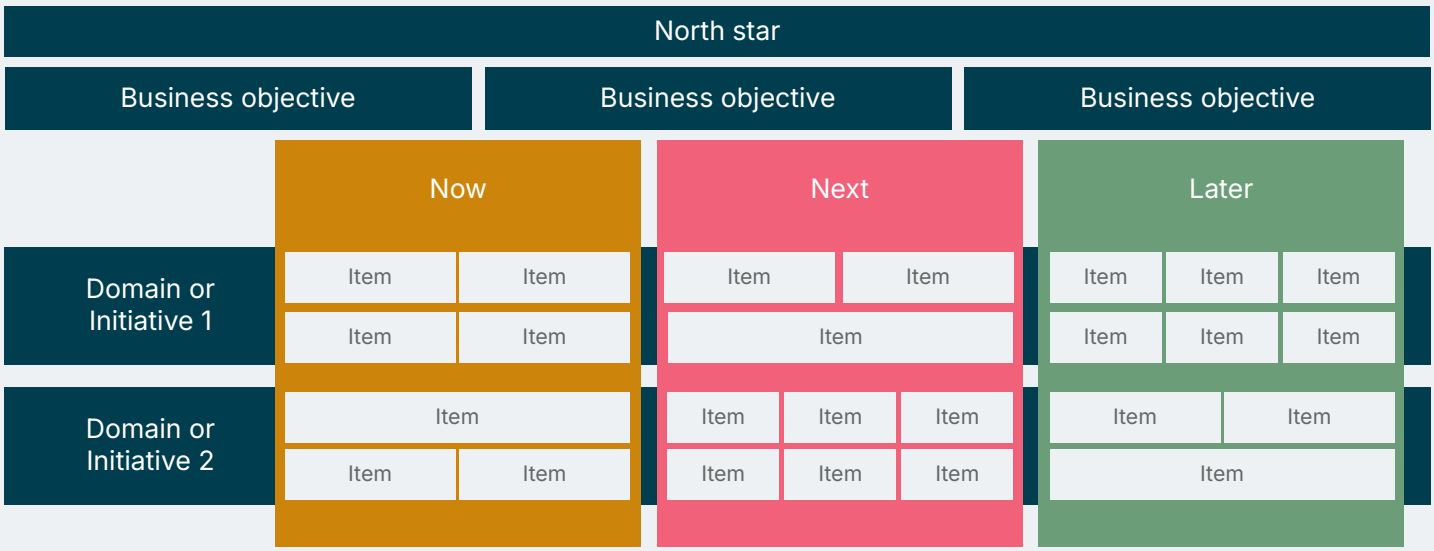
Architecture roadmap planning

1	**Start with the organization vision**	A clearly defined vision and a shared understanding of expectations for the future is the first step in creating an architecture roadmap. Ensure the organization vision is well understood. If anything is unclear, work with the appropriate people to break it down to ensure everyone has the clarity they need.
2	**Gather inputs from across the organization**	Have a clear understanding of the business goals and product aspirations. Map out where technology is now, what's working, what's not working, plus note down what's needed for the future to enable the organization to achieve and exceed their goals. It's best to use an online whiteboard for this. Record one idea per sticky. Where possible, write in terms of business capabilities. As you go, group stickies that are related to one another. Use colors to represent different dimensions to help keep track and manage the stickies. Gathering inputs upfront ensures you and your team have all of the necessary context, therefore enabling you to make better decisions.
3	**Create themes to focus outcomes**	A roadmap should speak to what you need to solve. What does the business need to achieve their goals? What are the customer needs? Now you have context and ideas, start grouping them into themes. Themes could be based on the business strategy, they could be high level customer needs, they could be have a system focus. Ideally, they clearly map to the vision and are easily understood.

Architecture roadmap planning

4	**Ruthlessly prioritize**	Frequently, organizations take on too much with seeming everything in progress at once. Sometimes just getting an MVP complete is a challenge. Prioritization helps you focus on the most important items. There are countless ways to prioritize. It's likely you will need a few rounds to get a view that is reasonable and has common agreement and buy-in. See next page for an example prioritization (bubble) diagram.
5	**Pull it all together to create your roadmap**	A roadmap tells a story, it is something you should gather round to have a conversation. It includes your north star, business objectives, high-level timeframes, items grouped by themes, and a simple now, next later view. Present your roadmap in the simplest form possible to make it easier to read, digest, and update. Roadmaps aren't set in stone once they are created; review and update your roadmap quarterly.

Roadmap presentation example



Example

Architecture roadmap planning

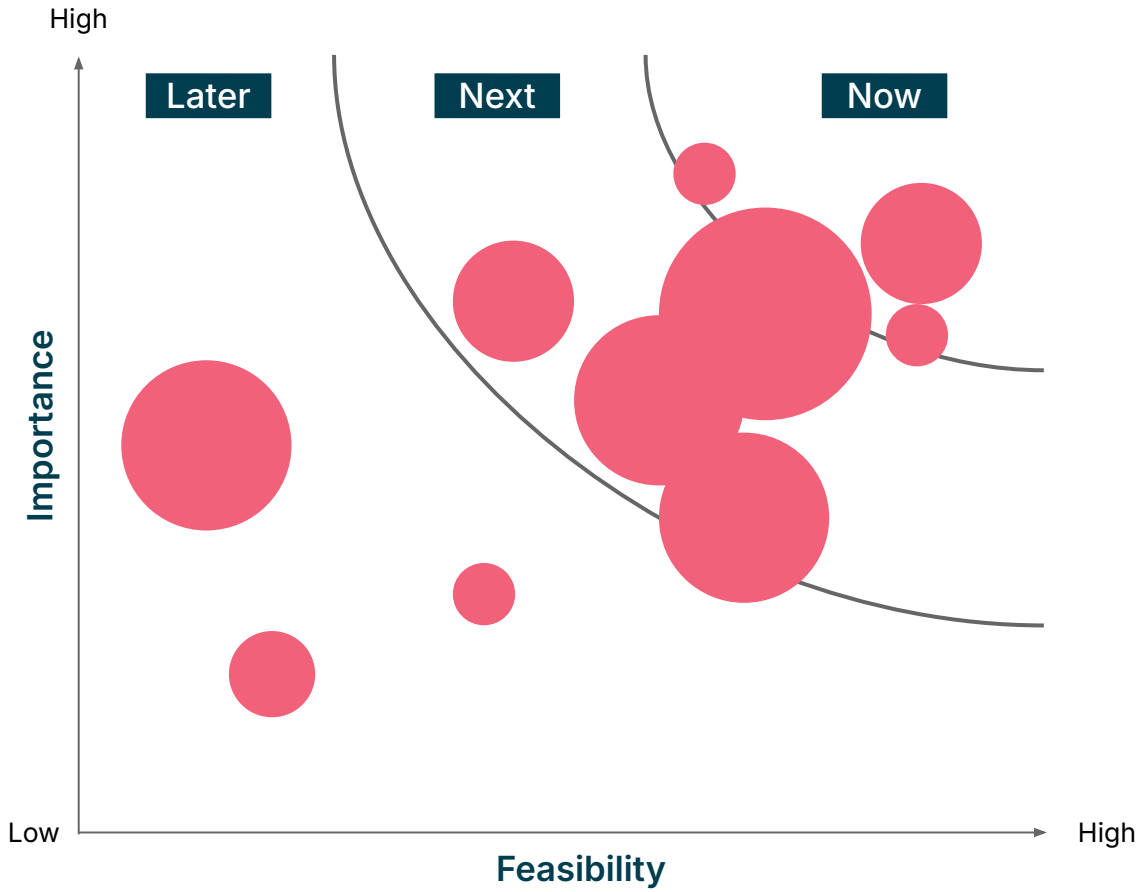
An example prioritization (bubble) diagram. The waves enable you to define now, next later. Circle size indicates effort.

Alternatives for x axis:

- Impact on achieving [vision/goal]
- Impact on business strategy

Alternatives for y axis:

- Customer value
- Business value



Play #22

Systems health assessment

A system health assessment will enable you to assess the health of all systems and to visually represent how operationally fit the organization is from a technical standpoint.

A system health assessment is a snapshot in time. It empowers architects, product and business stakeholders to make decisions as to where to invest effort and funds. It surfaces current state health for discussion within teams, and amongst technology leaders. A systems health assessment gives clear guidance to the organization on standards that must be met.

Use this play to:

- Understand the technical health of the organization
- Create visibility and a shared understanding of areas that pose a risk to the organization and its customers
- Help business stakeholders understand the health of systems and why investment is needed

Expert tip

Define the dimensions with the principal/lead developers to aid with buy-in and a sense of ownership and control.

Time required

2 hours

Who participates

Architects, principal/lead developers

How often

Every 6 months

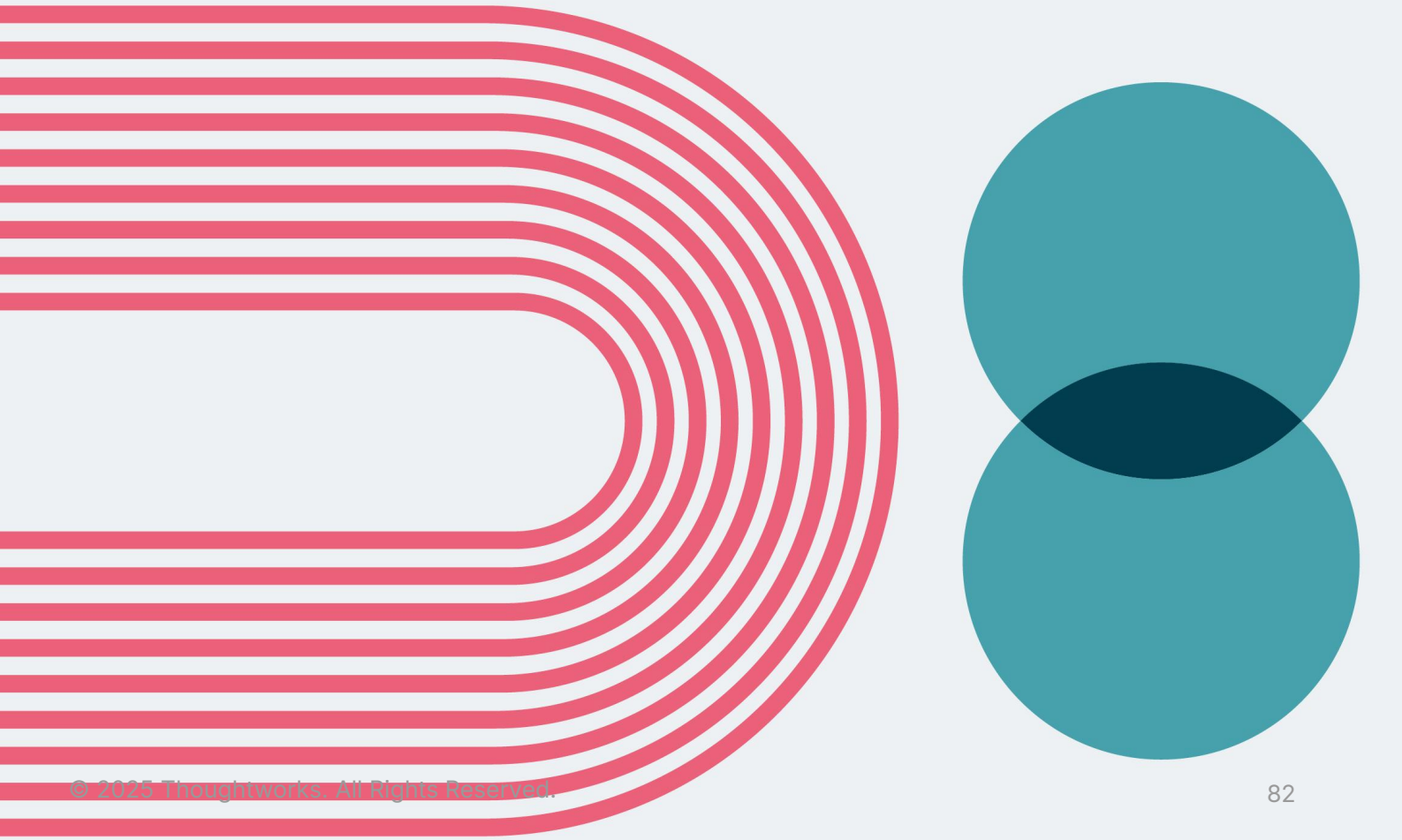
Systems health assessment

1	**Identify dimensions to be assessed**	A good starting point is the "-ilities", scalability, traceability, for example, along with security and data. As a group of architects, discuss what is important to the organization. Throw out ideas at a high level to start with. Draw inspiration from your architecture principles, organizational values and the business goals.
2	**Group and refine the dimensions**	Once you've brainstormed all dimensions, group and refine them. Some may be measures of a dimension and not a dimension in itself. Other may be repeated or very similar. Refine the wording so it makes sense to all.
3	**For each dimension, get specific on the measures**	Now you've got a list of dimensions, what is the measure for each. For example, if availability is a dimension, the measure could be zero downtime deployments. Another example is a Disaster Recovery plan. Is the measure simply that one exists? Or is the measure that one exists *and* it has been exercised at least every 6 months?
4	**Narrow down the dimensions**	At this stage, you probably have a very long and comprehensive list and you're wondering if there is a single system that will meet all of the dimensions. You need to add a touch of reality to the assessment and identify the low bar - what dimensions *must* be met for the system to be deemed healthy at a minimum? What would be the next layer after the low bar has been met? Separate out the 2, making sure it's clear what is required to be met to achieve the low bar.

Continued next page...

Systems health assessment

5	**Assess each system**	This can be done by sending a survey, creating a shared spreadsheet for owners to contribute to, or by having 1:1 conversations with system owners to ascertain the health of the system. Owners are technical owners, usually a principal or lead developer.
6	**Collate and present back the assessment**	Once you've gathered all of the data, visualise it and present it back. This might be for the organization as a whole, by domain or business unit or if you want to include the detail, at a system level. An example is provided on the next page. It's important to make sure actions to achieve the low bar are recorded in backlogs for each system.

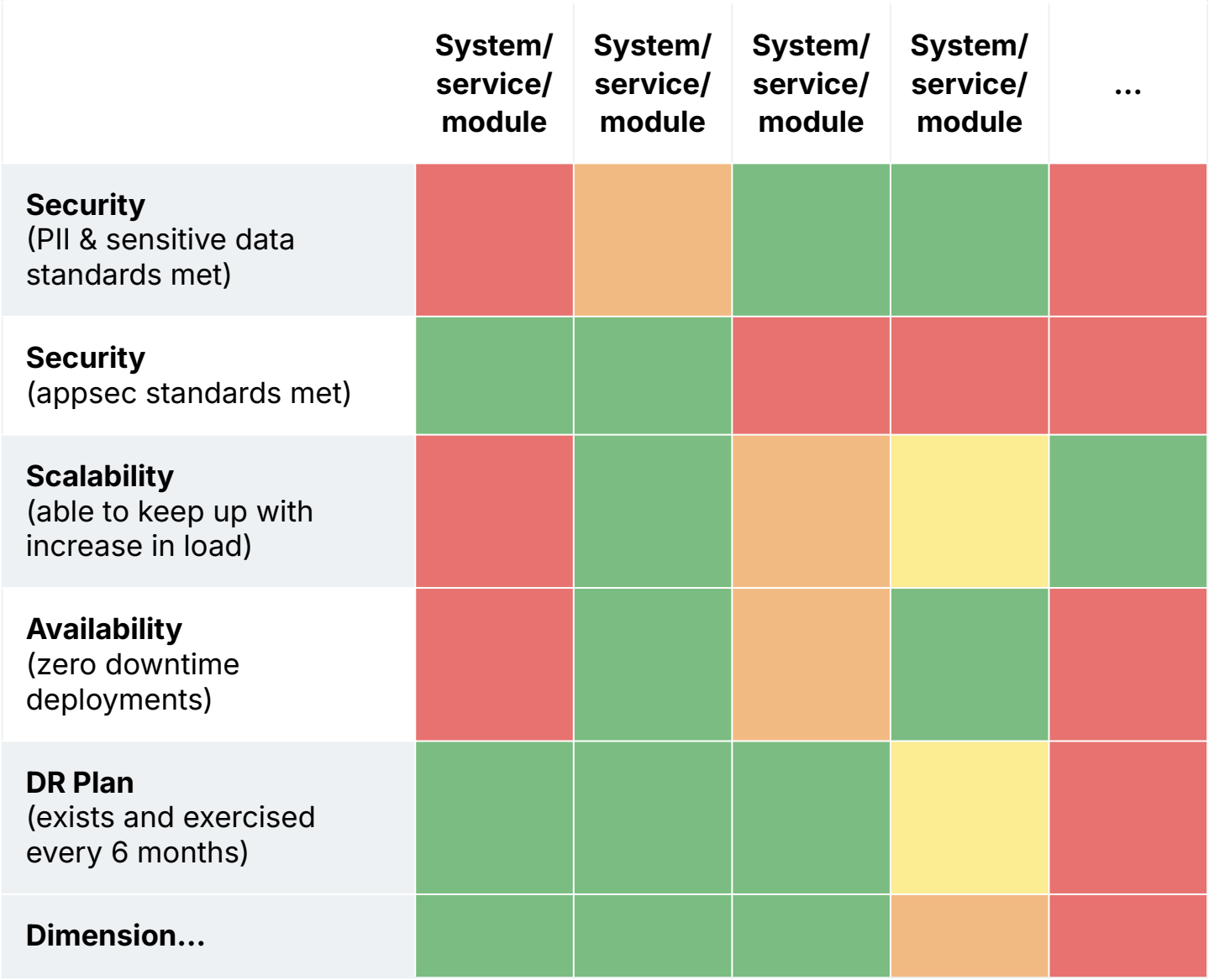


Example

Systems health assessment

The system health assessment provides a visual snapshot of how operationally fit the organization is, and highlights areas requiring urgent attention.

[Domain name] system health assessment



*Not sure can be used in cases where knowledge gaps exist and/or there is no ownership. Clear actions to identify an owner and to asses the system/service must be in place.

Play #23

Emergent capability mapping

Emergent capability mapping is a visual tool that helps organizations understand how their customer journeys, business capabilities, and technology intersect. It helps define products holistically, align investments with customer outcomes, and create a flexible, composable architecture.

Unlike traditional business capability maps that capture a current-state view, this play offers an evolving perspective on how your business platform should grow to meet future needs.

Emergent capability mapping is essential for organizations looking to transition to a product-oriented structure - helping to visualize their digital platform, connect investments to customer outcomes, and align both product and engineering perspectives.

Use this play to:

- Identify core business platform products and emerging capabilities
- Visualize and assess investments across each stage of the customer journeys
- Support the development of a composable architecture, with capabilities and services becoming modular and decoupled building blocks of the organization's product portfolio

Expert tip

Gain a complete picture and avoid siloed thinking by mapping business platforms organization-wide. Start with core customer journeys and build a portfolio of customer journeys over time by layering in secondary journeys. This keeps the approach manageable and scalable.

Time required

1 week - 4 weeks

Who participates

Architects, experience designers, business, product, technology representatives

How often

Every 12 months, or as as new core journeys are introduced

Key considerations

Emergent capability mapping

Emergent capability mapping is a strategic approach that focuses on identifying future business capabilities based on customer journeys. Keep these points in mind as you apply this method:

- **Alternate way to identify and prioritize capabilities**
  Unlike traditional capability mapping, this approach reprioritizes capabilities based on their relevance to customer journeys. It can reveal high-impact capabilities that might be overlooked in a static, business-unit-driven view and highlight those that need to evolve to support changing journeys. Without this, functionality may be misplaced, increasing coupling and technical fragility.
- **Emergent view through customer journeys**
  Identifying business capabilities through cross-business-unit customer journeys provides a dynamic perspective. It highlights intersections where reusable, scalable, and extensible capabilities can be developed.
- **Blank sheet approach**
  Start with a blank slate - set aside business units, domains, or team biases to uncover new opportunities. After creating an emergent capability map, compare with your existing capability map to identify key gaps, divergences, and opportunities for alignment and growth.
- **Breaking down silos**
  Many organizations face internal silos that prevent collaboration and efficiency. Mapping capabilities to cross-business-unit customer journeys exposes duplication and redirects wasted investments toward reusable capabilities that serve multiple areas.
- **Annual review**
  As customer journeys evolve, regularly reviewing your portfolio - ideally annually - ensures your capability map continues to surface new intersections and investment opportunities.

Emergent capability mapping

Do

- Start fresh without being constrained by your existing business structures. This will allow you to focus purely on the intersections of journeys and capabilities.
- Use customer journeys as the backbone of your map. This ensures your organization's capabilities align with customer needs and experiences.
- Prioritize strong intersections. These should become the foundation for your future business platform investments.
- Conduct regular reviews to keep your emergent capability map aligned with your business goals and evolving customer demands.

Don't

- Delve too deep. Emergent capability mapping typically identifies capabilities at the domain or subdomain level. This granularity helps create a more flexible, scalable architecture, as it allows for a detailed understanding of where intersections between journeys and capabilities exist.
- Focus solely on your current state. Avoid anchoring your decisions to legacy systems or team structures. Emergent capability mapping is about the future, not maintaining the past.
- Overlook duplication. Carefully examine intersections to avoid investing in redundant capabilities that do not provide strategic advantage.

Emergent capability mapping

In some cases, after completing emergent capability mapping, you may find few or no clear intersections between customer journeys and business capabilities. While this may seem like a setback, it simply means a business platform is not a priority. However, it presents an opportunity to prepare for the future.

- **Focus on modular, composable services**
  Even without strong intersections, your business still needs services to support customer journeys. Rather than building a unified platform, focus on modular, composable services that are decoupled and shareable. These services complete journeys today and, as reuse grows, may later evolve into capabilities managed by a dedicated team.

  Alternatively, you may still be able to define a platform by grouping services that share functional, technical, or interaction affinity into a distinct boundary. This does not require rigid coupling - because services remain composable, they can be rearranged or evolved over time as new opportunities emerge.
- **Prepare for future growth**
  Think of this as laying a foundation. If your business capabilities and services are tightly coupled, adding customer journeys, or evolving architecture, will be cumbersome - akin to open-heart surgery on your systems. Decoupling and modularizing now creates flexibility to integrate future journeys seamlessly.
- **Practical steps for decoupling**
  Identify where capabilities can become independent services. Focus on decoupling today for future reusability, rather than forcing reuse too early, which increases coupling at the code, library, or messaging levels. This may involve adopting microservices, creating APIs, or defining clear service boundaries for long-term flexibility.
- **Stay agile**
  Regularly review customer journeys and capabilities. As the organization grows, new opportunities for reuse and platform development may arise. With decoupled, composable services in place, you'll be well-positioned to pivot and build an integrated business platform when the time is right.

“If there are no strong intersections, then you are simply creating individual services for a specific journey. Over time, these services may be assimilated into capabilities and a platform.”

Omar Bashir, Technology Director, APAC BFSI, Thoughtworks

Emergent capability mapping

There are 4 phases to emergent capability mapping:

Phase 1. Prepare

1	**Identify collaborators and contributors**	Emergent capability mapping requires cross-functional input. Identify key stakeholders from across the organization, including those closest to the customer, business process experts, and those familiar with system and service design. A diverse group will provide a broader, more accurate perspective.
2	**Share the what and the why**	Clearly communicate the purpose and benefits of emergent capability mapping. Ensure everyone understands that the goal is to identify intersections between customer journeys and business capabilities, reducing duplication and highlighting reusable capabilities.
3	**Establish timeframes**	Establish clear milestones and timelines. Setting a phased approach helps manage expectations and keeps the process on track.
4	**Start small**	Begin with a limited set of high-impact customer journeys to show early wins and maintain momentum.
5	**Clarify definitions**	Ensure a shared understanding of key concepts, including what constitutes a capability and the intended outcomes of the mapping process.

Emergent capability mapping

Phase 2. Build the foundations

6	**Identify customer journeys**	List customer journeys, categorizing them into core (high-impact, frequent interactions) and secondary (lower-impact, less frequent interactions). At this stage, focus on naming the journeys rather than detailed mapping.
7	**Collect existing artifacts**	Many organizations already have some relevant artifacts in place. Gather existing resources such as customer personas, journey maps, or process documentation to avoid duplication of effort.
8	**Identify customers**	Clarify who the customer is for each journey. Use existing personas or create new ones where needed.
9	**Ensure consistency**	Select a standard journey map format to maintain consistent levels of granularity across all journeys. This consistency is crucial for later overlaying journeys against business capabilities.
10	**Refine existing journeys and create missing ones**	For each of the core journeys identified, create (or refine) a journey map following the standardized format to enable effective comparison and integration.
11	**Identify required supporting capabilities**	For each core journey, identify the key business capabilities needed to support it. Focus first on individual journeys to avoid overwhelming teams. Then, recognize common functions or processes shared across multiple journeys - these often signal reusable capabilities. Note: The broader review of how journeys interact and share capabilities is covered in Phase 3.

Emergent capability mapping

Phase 3. Identify points of intersection

12	**Overlay customer journeys and capabilities**	Lay out customer journeys alongside business capabilities to identify where multiple journeys rely on the same capability. These intersections indicate potential areas of reuse.
13	**Highlight strong intersections**	Emphasize capabilities that serve multiple journeys. These high-value intersections are candidates for increased investment.
14	**Assess the state of capabilities**	Determine whether shared capabilities are already composable and reusable or if they exist in multiple, fragmented versions. Identifying duplication and gaps will inform decisions about where to consolidate or improve.

Phase 4. Make strategic decisions

15	**Determine the need for a business platform**	Once you've identified the intersections, assess whether the strength and frequency of intersections justify investing in a business platform. If shared capabilities are minimal or weakly connected, a platform may not be necessary at this stage.
16	**Assess platform scope**	If there is a need for a platform, decide whether a single platform is sufficient or whether multiple platforms are required to support different domains. For example, strong intersections in some areas might support a business platform, while others may not. Platforms should have functional affinity and clear boundaries to avoid unnecessary complexity.

Emergent capability mapping

17	**Form long-running product teams around core capabilities**	For capabilities with strong intersections, consider forming dedicated product teams to manage and evolve them. These teams should: • Treat the capability as a product, offering reusable services to the organization • Be led by product managers responsible for its value and adoption • Be funded proportionally from the technology budget allocated for customer journeys, ensuring sustainability and alignment. Funding the teams: These long-lived product teams can be funded through the same technology budget allocated for customer journeys. Since the customer journeys rely on underlying business capabilities, a proportionate share of that budget should be directed to the teams managing those capabilities. This ensures sustainable funding and alignment between the customer journey investments and the core capabilities supporting them.
18	**Identify and eliminate wasted investments**	Look for redundant capabilities maintained by multiple teams. Redirect teams toward building composable, shareable services that eliminate inefficiencies.
19	**Create a roadmap for future growth**	Even if a platform is not needed today, begin building modular, decoupled, and composable services that can be integrated into a future platform. This ensures flexibility as new customer journeys emerge.

Play #24

Business platform strategy

A business platform strategy is the roadmap to leveraging business platforms for achieving your organization's objectives. These platforms act as compelling internal products, offering core business capabilities as self-service solutions to the rest of the organization.

Just like an API strategy, the absence of a business platform strategy results in information, data, and capability silos. This creates a fragmented view of the business and hinders collaboration. Additionally, reliance on outdated processes makes adapting to change difficult, limiting your ability to compete in a dynamic environment.

A business platform strategy goes beyond simply using technology. It emphasizes interoperability and enhanced agility. By breaking down data and capability silos, it allows information to flow freely, fosters collaboration, and delivers a unified view of the business. This "platform thinking" approach facilitates quicker decision-making and smoother integration of new technologies, ultimately enhancing your organization's ability to adapt to change effectively.

Use this play to:

- Formulate a business platform strategy
- Accelerate digital product delivery by reducing friction between teams and streamlining access to core business capabilities

Expert tip

A business platform must be seen as a product. A product-centric approach ultimately leads to a more successful and sustainable business platform that drives user adoption and delivers a strong competitive advantage.

Time required

2 - 4 weeks

Who participates

Architects, business, product and technology leaders and executives

How often

Review every 6 months

“A digital platform is a foundation of self-service APIs, tools, services, knowledge and support which are arranged as a compelling internal product. Autonomous delivery teams can make use of the platform to deliver product features at a higher pace, with reduced coordination.”

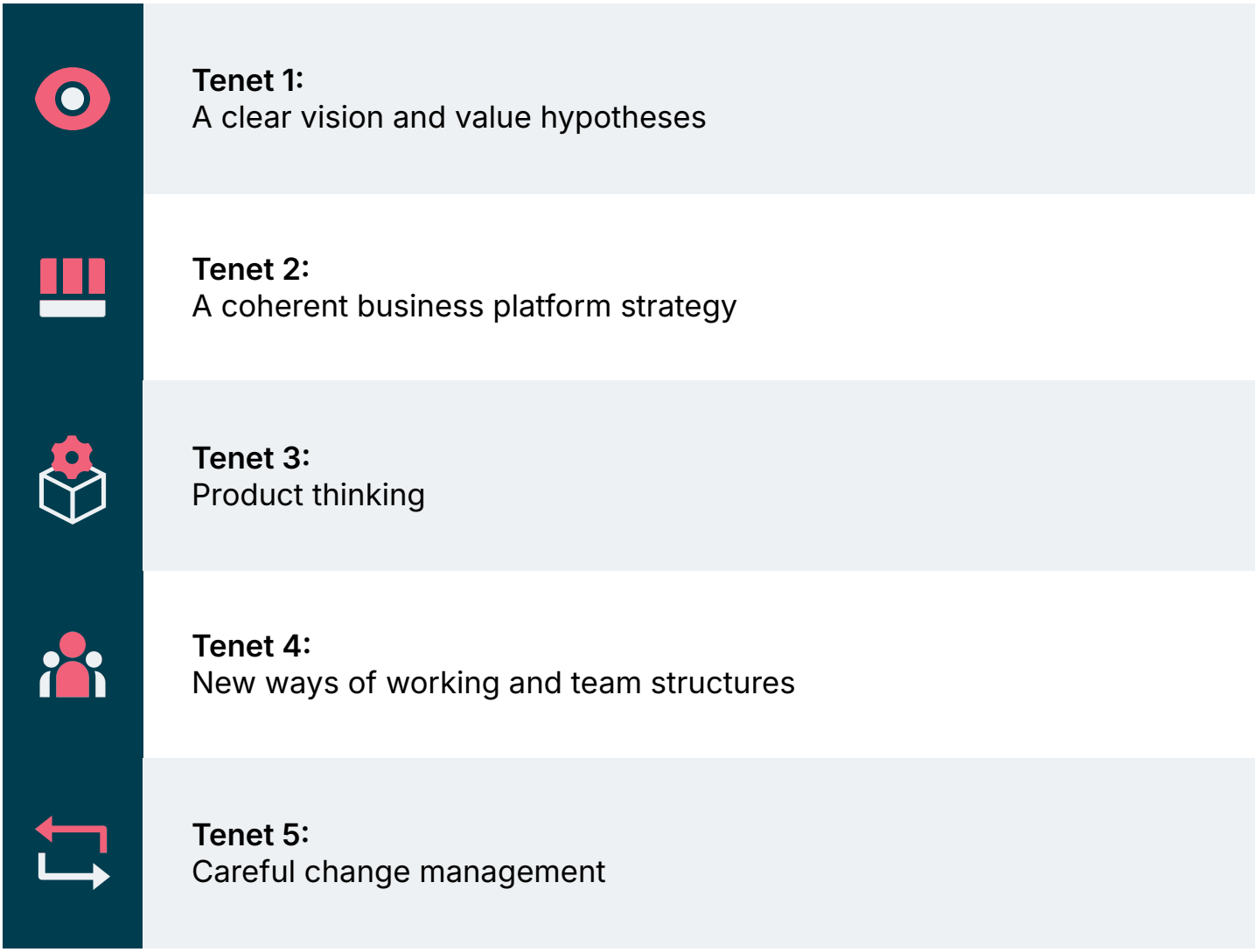
Evan Bottcher

Business platform strategy

The five tenets of platform success

The potential value of platform thinking alongside platform building is undeniable. But, to realize that value, you need a strategy that looks beyond building the platform, one that helps your entire organization evolve to get the most from whatever platforms you choose to create.

Here are five tenets that are instrumental to the success of any platform journey:



Tenet 1:
A clear vision and value hypotheses

Tenet 2:
A coherent business platform strategy

Tenet 3:
Product thinking

Tenet 4:
New ways of working and team structures

Tenet 5:
Careful change management

Source

Business platform strategy

1	**Identify collaborators and contributors**	A business platform strategy should not be created in isolation. Identify key folks across the organization who can collaborate with you and/or contribute to the strategy.
2	**Get clear on the purpose, vision and goals**	As a group, start by getting clear on the organization's current purpose, vision, goals and business strategy. Additionally, if technology, platform, and product strategies exists in the organization, ensure there is a common understanding and the vision for each function shared. Finally, discuss the purpose of the business platform strategy, why the organization needs it and why now.
3	**Get a holistic capability-view of the organization**	Create a picture of the organization today in terms of product, business and technology capabilities. Run Play #23 - Emergent capability mapping to create a view of the organization and identify core business capabilities.
4	**Understand the market in which the organization operates**	This applies both internally and externally. What is going on in the market that is impacting or could impact the business. What is the state of the internal environment? What factors influence or impact product management, technology, architecture, and business direction? Play #19 - Wardley Mapping, can help with this.
5	**Perform capability gap analysis**	What capabilities are required to execute the business goals and vision? Where are capabilities duplicated? What gaps exist today? Run Play #18 - Capability Gap Analysis.
6	**Synthesize and summarize current state**	Clearly articulate the current state and provide a short summary. Leverage the Business Platform Map - It visually provides a concise summary of core customer journeys, business processes, and capabilities.

Business platform strategy

7	**Socialize and refine**	Share your synthesized view of the current state, using the business platform map as a reference. This socialization process allows for valuable input and ensures everyone is aligned on the big picture. Encourage feedback and additional insights from stakeholders across the organization (beyond your group of collaborators and contributors). This collaborative approach helps to refine your understanding and create a more comprehensive picture.
8	**Chart the Course: vision, problems and future state**	Brainstorm business platform goals to enable the organization to realize their vision. Identify problems the organization is facing today. Think about what needs to change and what needs to exist that doesn't currently. And finally, describe the future state: Where does the organization need to be in 3-5 years?
9	**Assess the landscape**	Assess technical debt and alignment challenges. Consider if the organization is suffering from innovation debt, technical debt, or misaligned talent and organizational structures.
10	**Identify core business capabilities**	Use outputs from steps 3-6 to determine the business capabilities needed to support the business today and in the future. Run Play #16 - Capability Assessment, to identify which capabilities create competitive advantage, provide strategic support or are business necessity.
11	**Confirm the goals**	Take outputs from steps 8-10 and define/refine the big goals. Ensure everyone in the group is aligned and the language used is understood by all. These become your strategy items.

Business platform strategy

<table>
<tr><td>12</td><td>Identify actions</td><td>Identify actionable steps to achieve each goal (strategy) and address the problem/challenge being faced today.</td></tr>
<tr><td>13</td><td>Write the strategy</td><td>Form each goal, aligned with problems faced today and actionable steps into a strategy item. An example of how to structure each strategy item is included here:<table><tr><td>Strategy title</td><td>Short descriptive statement.</td></tr><tr><td>What</td><td>What the strategy is and what needs to be true.</td></tr><tr><td>Current state</td><td>1 - 2 sentences to describe the current state.</td></tr><tr><td>Benefits</td><td>1 - 2 sentences to describe the benefits that will be realized in addressing this strategy item. Include both quantitative (e.g., increased revenue) and qualitative (e.g., improved developer experience) benefits.</td></tr><tr><td>How to get there</td><td>Actionable steps to execute this strategy item (short and specific bullet points).</td></tr></table></td></tr>
<tr><td>14</td><td>Gather feedback and refine</td><td>Socialize the initial strategy draft for feedback with a broad range of stakeholders across the organization (beyond your group of collaborators and contributors). This collaborative approach ensures continued buy-in and alignment.</td></tr>
<tr><td>15</td><td>Incorporate feedback and validate</td><td>Incorporate valuable feedback into the strategy and address any concerns raised.
Conduct a final validation session with key stakeholders to confirm alignment.</td></tr>
</table>

Business platform strategy

16	**Create a roadmap**	For each actionable step, discuss and agree on its priority, ultimately placing it into "Now," "Next," and "Later" buckets. Start by **identifying dependencies.** Understand how the actionable steps rely on each other. **Next,** determine the factors that will guide your prioritization. This could involve value, impact, or risk reduction potential as a few examples. Finally, with dependencies and criteria established, prioritize each step into your "Now," "Next," and "Later" buckets and visualize into a roadmap. This visual roadmap will provide a clear picture of the execution plan. See Play #21 for a practical example of creating a roadmap.
17	**Tie it all together and share broadly**	Develop a compelling narrative that highlights the strategic importance of the business platform strategy. Emphasize the value proposition of the strategy for different stakeholders, including customers, partners, and internal departments and teams. To be successful, the entire organization must be behind the strategy. Share the strategy and share background on platforms and platform thinking. Give employees the full picture. They too need to know where the business is today, what is holding it back, what is propelling it forward and how this business platform strategy fits in the picture.
18	**Enable teams to execute the strategy**	Align small teams around business capabilities (give them ownership) and empower the teams to deliver these capabilities to one another taking a product-centric approach.
19	**Measure success**	Identify key metrics to track the progress towards achieving the strategic goals. Note that consistent deviations from the plan might indicate a need for revision.

Business platform strategy

Characteristics to consider when defining a business platform strategy

Ambitious leadership vision - What is the executive vision for becoming a platform-enabled enterprise?

Business strategy alignment - How well does the platform strategy support business goals?

Multidimensional - What do we need to be successful in platform development beyond the technology itself?

Lean, adaptive, evolutionary - How well does the strategy adapt to change and evolve over time?

Outcomes driven - Is the platform strategy both aspirational and grounded in observable outcomes? Concrete and actionable - Does the strategic roadmap contain clearly defined initiatives?

Scalable and sustainable - Does the strategy lend itself to developing foundational tools, process and people capabilities such that the organization can evolve and scale to new problems? Beware “tactical strategy”.

1. Continuously reevaluate the businesses needs and evolve the platform to support them

2. Make product thinking a priority

3. Make sure your teams are well aligned with your platform goals

Play #25

API Strategy

A strategy is the purposeful investment of resources in advance to affect a particular outcome, often competitive in nature.

Without a clear API strategy, organizations can struggle with siloed data, slow innovation, and missed opportunities for collaboration. Conversely, an effective API strategy serves as a solution to these obstacles.

By strategically planning for APIs, businesses can break down information barriers, streamline development, and unlock the full potential of APIs and capabilities for competitive advantage.

An API strategy empowers organizations to prioritize the design, development, and integration of APIs as integral components of their operations. This recognition of the strategic value of APIs fosters a developer-centric approach and enables organizations to leverage APIs as drivers for growth, innovation, and collaboration both within and beyond their organizational boundaries.

Use this play to:

- Unlock the power of APIs with a strategic approach
- Describe how you'll achieve your end (goals) with your means (resources) available
- Design a plan for how the organization will build, use and govern APIs

Expert tip

Collaborate from the start! Transparency and early buy-in ensure smooth execution. Never create a strategy in a silo.

Time required

2 - 4 weeks

Who participates

Architects, technology leaders, principal developers, delivery teams

How often

Review every 6 months

Example

API strategy

Elements of an API strategy

Introduction - purpose, why, and how it ties into the organization vision	Why the organization needs an API strategy
	The purpose of the strategy
	How the strategy supports the organization to realize their vision
Strategy - long term plan designed to achieve objectives	Overview
	Current state description, future state description and actionable steps
	Roadmap
Governance - principles and guidelines that supplement the strategy	API types
	API design principles
	Interaction patterns
	Discoverability, usability and documentation standards
	Security standards
	API lifecycle management

API strategy

1	**Get clear on the purpose, vision and goals**	Start by getting clear on the organization's purpose, vision, goals and business strategy. If a technology or platform strategy exists in the organization, ensure there is a common understanding. Additionally, discuss the types of APIs in-use in the organization today, for example internal, external or both. Finally, discuss the purpose of the API strategy, why the organization needs it and why now.
2	**Define the elements of the strategy**	Define the elements that will make up the API strategy (see example previous page). Include a short definition of each element and make sure the group is in agreement.
3	**Understand the current state**	Map out, describe and discuss the current state in regards to APIs. Consider what challenges the organization and technology teams face today? What are the common issues being raised? What is needed (from an API perspective) to ensure the organization can realize their vision? What are the gaps that exist today? It is best to go wide and hear from folks across the organization, not just those crafting the strategy. This includes speaking with developers both internally and externally (if you are publishing APIs for external consumption). You can achieve this by running a number of workshops to elicit insights, feedback and recommendations from folks.
4	**Synthesise and socialise the current state**	Synthesise the current state. Clearly articulate the current state and problems faced today. Socialize these across the organization to check you've understood and you've allowed for any further commentary.

API strategy

<table>
<tr><td>5</td><td>Brainstorm goals and the future state</td><td>Brainstorm goals to address the problems. What needs to happen to ensure the organization can realize their vision? Where would you like to see the organization (from an API perspective) in 3 - 5 years? Describe the future state for APIs within the organization.</td></tr>
<tr><td>6</td><td>Confirm goals</td><td>Discuss and align on the big goals - these will become your strategy points. Aim to have 3 - 4 points.

Each goal should have at least 1 problem mapped to it.</td></tr>
<tr><td>7</td><td>Identify actions</td><td>Identify actionable steps to achieve the goal (strategy) and address the problems being faced today.</td></tr>
<tr><td>8</td><td>Write the strategy</td><td>Form each goal, aligned with problems faced today and actionable steps into a strategy item. An example of how to structure each strategy item is included here:

<table>
<tr><td>Strategy title</td><td>Short descriptive statement</td></tr>
<tr><td>What</td><td>What the strategy is and what needs to be true</td></tr>
<tr><td>Current state</td><td>1 - 2 sentences to describe the current state</td></tr>
<tr><td>Benefits</td><td>1 - 2 sentences to describe the benefits that will be realized in addressing this strategy item. Include both quantitative (e.g., increased revenue) and qualitative (e.g., improved developer experience) benefits.</td></tr>
<tr><td>How to get there</td><td>Actionable steps to execute this strategy item (short and specific bullet points)</td></tr>
</table>
</td></tr>
</table>

API strategy

9	**Socialize the first draft for feedback**	Socialize the first draft of the strategy for feedback with a broad range of groups across the organization. This helps continue to maintain buy-in and alignment. Work in feedback and sense-check with key people.
10	**Create a roadmap**	For each actionable step, discuss and agree on the priority of each and place them into “Now”, “Next”, and “Later” buckets. Start by mapping out and understanding dependencies between each actionable step. Once you’ve done this, determine your criteria for prioritization, for example value, impact, risk reduction etc. Prioritize and visualize into a roadmap.
11	**Tie it all together and share broadly**	Add your introduction and overview sections. Develop a compelling narrative that highlights the strategic importance of APIs in achieving organizational objectives. Emphasize the value proposition of the API strategy for different stakeholders, including customers, partners, and internal teams. You will have likely covered some, if not most of the governance elements (as described here) as actionable steps in the strategy - if they aren’t covered, be sure to note down actions to create each supplemental document. Share the strategy with the organization making sure it is discoverable and accessible by all.
12	**Measure success**	Identify key metrics to track the success of your API strategy.

Example

API strategy

Example strategy item

[Title]
Strategy: Streamline API Development with Lightweight Governance

[What]
To build trust and improve data sharing, we're implementing flexible API governance. This includes clear standards, processes, and best practices for API development and management. This will ensure consistent, predictable APIs, which in turn fosters better collaboration, clearer visibility into data, and increased trust among developers.

[Current state]
Right now, data sharing is hampered by issues like unclear data origin (lineage), unclear ownership of data and APIs, inconsistent API designs, and a focus on quick fixes over long-term solutions. This leads to information overload, repetitive tasks, and difficulty using APIs to share capabilities. As a result, teams struggle to focus on building innovative features.

[Benefits]
By addressing these challenges, we can:

- Reduce risks by implementing strong API practices
- Improve collaboration, trust, and data visibility within the organization
- Free up developer time by minimizing information overload and repetitive tasks, allowing them to focus on innovation

[Actionable steps]
How to get there:

- **API Style Guide** - create a clear set of guidelines for API development to ensure consistency across teams
- **API Design Principles** - define core principles for API development to align with our overall goals
- **Improve API Documentation** - equip teams with tools to create user-friendly API documentation, making them easier to find, understand, and use
- **API Governance Policies** - establish clear rules for API ownership, access control, and compliance

Measure success

API strategy

As Kirsten Hunter calls out in Irresistible APIs, "**...think about how your API is enhancing the goals of the company as a whole rather than determining how many people have begun to integrate with your system.**"

Identifying key metrics to track the success of your API strategy is just as important as creating the strategy itself. Here are some lenses to consider when identifying key metrics.

Effectiveness of the strategy

The purpose of an API strategy is to leverage APIs as drivers for growth, innovation, and collaboration and foster a developer-centric approach. This should lead to creating APIs that are resilient, secure, and capable of rapid change at scale.

To measure the effectiveness of the strategy consider using look-back metrics such as the DORA metrics; Deployment Frequency, Lead Time for Changes, Change Failure Rate, Time to Restore Service.

Effectiveness of your APIs

Consider how your API enables business outcomes.

- **Monetization** is appropriate if the API is your main product i.e. Twilio, however inappropriate otherwise because it will compete with your main product.
- **Usage** is appropriate if your business is focussed on interactions i.e. LinkedIn and Facebook where a compounding effect happens by encouraging writes to an activity feed.
- **Partner retention** is a great lens when your focus on providing analytics to your partners. For example white label credit cards and Expedia.
- **Market dominance** is a lens Netflix applied with their device oriented APIs. For example, the number of devices Netflix is used on.

Measurement lens

Consider which lens your metrics will be viewed as each will have a different need. For example, a CFO lens will be different to that of a DevOps lens.

Execution and Governance

Play #27

Path to production

Path to production is a visualisation of all the steps your product takes from the initial problem or user ask, through to the hands of a developer, all the way into production and lastly onto end-of-life.

Value stream mapping is a lean management tool used in the manufacturing industry that helps visualise the steps from product idea, to delivery, to customers and for this reason is used as the framework to identify and define a path to production.

There will generally be three main steps in creating a path to production; capture the current state; discuss and analyze the current state before defining an ideal future state; create a roadmap and execution plan to get from the current state to the desired future state.

A path to production should have its own outcomes and measures of success so if the path to production has to be adapted, it still delivers the outcomes.

Use this play to:

- Identify bottlenecks and unnecessary rework
- Improve processes and throughput
- Provide the foundation for rapid, reliable software delivery

Expert tip

The path to production should be considered as a whole, as a system, not as independent steps. The goal is to optimize the end-to-end flow, and this requires both a holistic analysis of the system and deep dives into those steps.

Before running this play, make sure the architects have good organizational context.

Time required

5 hours over 3 meetings

Who participates

Architects, engineering teams

How often

As needed

Path to production

There are 4 steps in the Path to Production Process:

Prepare → **Map the current state** → **Design the future state** → **Build a roadmap**

Prepare

1	**Bound the process**	Bound the process. Decide what the limits of your map will be. Are you taking a holistic, end-to-end approach, starting at the user ask? Or narrowing down to focus on code checked-in through to production delivered?
2	**Identify your stakeholders**	Identify your stakeholders and who is needed to map out the detail. If you're taking a narrow approach, developers, QAs, ops and security should be involved. If you are taking a holistic approach and looking at the end-to-end flow, you'll need to widen your participant net to include Product Managers, Data Engineers and technology providers, to name a few.
3	**Prepare a working space**	Prepare a room with ~4 metres of wall or whiteboard space, or prepare an online collaborative space such as a Mural board.

Path to production

Map the current state

1	**Introduce path to production mapping**	State the purpose and desired outcome. Create a safe space; let participants know there is no judgement or blame - the purpose is simply to understand the process and how well it works. Issues and waste will be uncovered, and enable participants to better uncover root causes.
2	**Start simple**	Capture the first activity and the last. This reiterates the bounds in which you're working and will allow participants to ease into the activity.
3	**Fill in the details; don't get hung up on perfection**	Start filling out the map, but don't strive for perfection. If it's a scribble on a whiteboard, or a mass of post-its, that's ok. If it's approximate duration instead of exact, down-to-the-millisecond, duration that's ok too.
4	**Stick to the problem**	Encourage the group to keep focus on the problems and as-is process. Avoid going into solution mode at this point.
5	**Deep dive as needed**	For example, if developers are branching, capture more detail on branching.
6	**Ask questions to draw out the detail**	Ask questions to draw out detail on how materials and information flow, reliability, pain points, duration and definition of done for each activity. Additionally, capture tools required and roles/people involved for each activity.
7	**Share**	Once you feel you've captured the right level of detail and have a solid understanding of the current process and how well it works, share it with the group and relevant stakeholders after the session. Make sure everyone is on the same page and ask for any further commentary.

Path to production

Design the future state

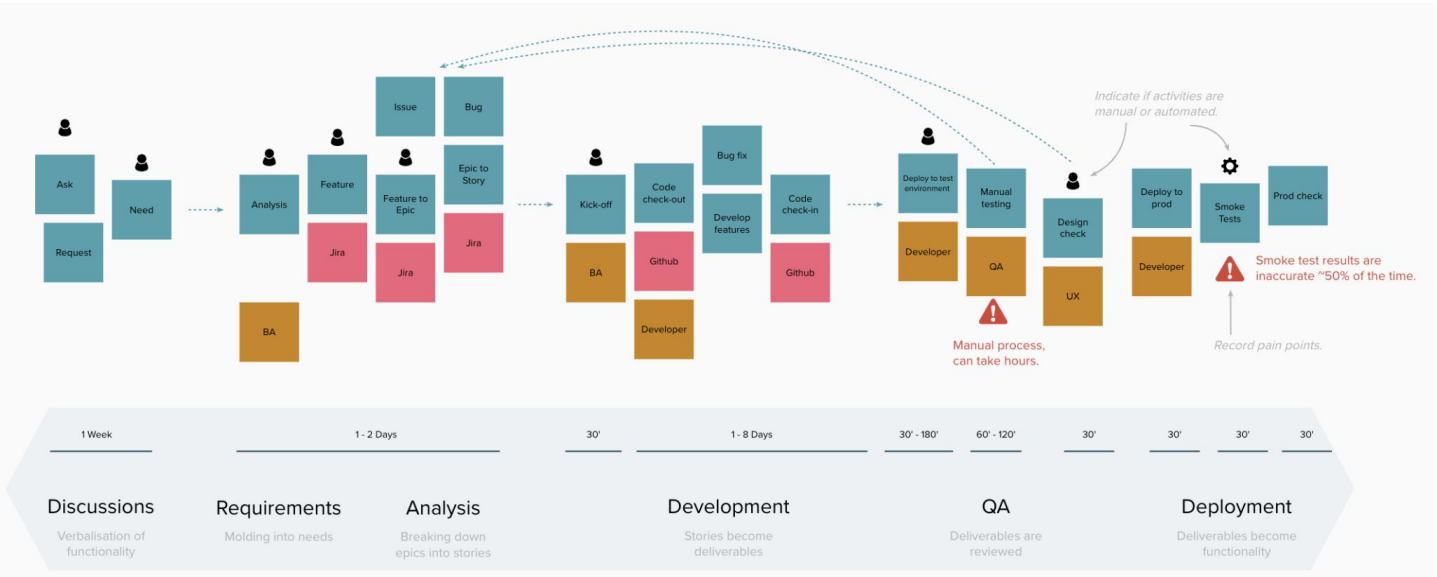
1	**Prepare**	Much like mapping out the current state, you'll need to prepare a space in which to work and identify stakeholders required to create a representation of what the path to production should look like.
2	**Understand constraints**	Before you start the activity, understand the constraints you'll be working with. The future state needs to be realistic and achievable within a specified timeframe. If you're designing a future state that will take 3 years to achieve, it's likely nothing will happen. When do stakeholders want to see a future state realized? What are the people and technology bounds to work within?
3	**Introduce the activity**	State the purpose and desired outcome. Share the current state; ensure people can easily refer to it. Make known constraints within which you're designing the future state.
4	**Identify customers**	Who are your customers? Identify who the customers are for each step. Capture their requirements.
5	**Look for waste**	What isn't adding value and can be removed? What steps can be improved? Where is the wait time?
6	**Find the repetition**	Find the repetition. Are there repetitive activities that can be automated? Can redundant activities be merged?
7	**Review and Share**	Walk through it end-to-end with the group to make sure everyone is on the same page and in agreement. Share it with stakeholders and anyone who wasn't able to attend the session. Write a high-level list of recommendations to accompany the future state document.

Path to production

Build a roadmap

1	**Prepare for execution**	The final stage is to prepare for execution. A high-level roadmap from the current state to the future state is produced. This roadmap should correspond to about three months of execution. This is enough time to produce concrete improvement but not so long that it becomes speculative. A lot will change in three months. The roadmap should include rough resourcing, governance and organizational responsibilities/commitments.

Path to production example



Play #29

Fitness functions

Being an Architect is all about making significant technical decisions. Architectural decisions are difficult to make and have significant ramifications on the organization, both positive and negative. We use Fitness Functions to provide feedback on how our decisions are contributing to meeting architectural requirements.

Fitness Functions are (usually) automated checks that help verify how well the architecture is supporting various requirements like security, performance and availability. These can be based on metrics collected from production, results from simulated transactions or static code checks.

For example, if Chaos Engineering is chosen (i.e., an architectural decision) to improve overall fault-tolerance and robustness, tracking the number of production issues caused by component failure before / after the introduction of Chaos Engineering will support whether this decision was valuable or not.

Use this play to:

- Build hypotheses behind architectural decisions
- Codify experiments to test these hypotheses
- Monitor the success of these experiments
- Adjust decision making as appropriate

Expert tip

Fitness functions only make sense if they are continually monitored and actioned - otherwise they are not different to normal broken tests: adding waste, not value.

WIP Limit the number of fitness functions being used at any time.

Time required

60 minutes

Who participates

Architects

Implementation

1 week

Fitness functions

Fitness Functions come in many shapes and sizes. It is useful to understand these categories as they describe a pattern language for discussing the functions with various stakeholders. For example, "I think we need an automated, atomic, continual function to ensure the new cache settings don't negatively impact our throughput."

Atomic versus Holistic	**Atomic** functions test a single architectural concern (e.g., performance), whilst **Holistic** functions test a combination of concerns, especially where there is a concern that natural tension between the concerns might result in a zero-sum game (e.g., usability and security).
Triggered versus Continual	**Triggered** functions execute in response to an event (e.g., a CI build). **Continual** functions are constantly running and gathering data from the system. Continual functions often use production metrics like operational logs as their primary input.
Static versus Dynamic	**Static** functions include a known and constant definition of "success" (e.g., maximum response times of 200ms to all GET requests). **Dynamic** functions vary the definition of success based on external factors (e.g., maximum response times of 200ms for the 95th percentile of GET requests at <= 200% of expected maximum load).
Automated versus Manual	**Automated** functions can be codified as automated tests and executed with no human intervention. **Manual** functions require some form of human interaction to execute. Some Manual functions might require 100% human support, either due to lack of automation maturity or inherent non-reproducibility of the test.

Fitness functions

Temporal	**Temporal** functions are functions that only make sense to evaluate at key time periods (e.g., known peak load, business critical events like Black Friday in the retail sector).
Intentional versus Emergent	**Intentional** functions are identified at/before the implementation of the architectural decisions they are supporting. **Emergent** functions are identified in response to concern in running software. This is a concern that wasn't identified at the time the relevant architectural decisions were being implemented.
Domain Specific	**Domain Specific** functions focus on areas specific to the problem domain in question. Such functions in social media domains might focus on data privacy. Similar functions in investment banking would likely cover throughput and performance. Source: https://evolutionaryarchitecture.com/

Some requirements are extremely hard to automate into fitness functions. Resist the urge to only test what can be automated. Manual fitness functions testing key decisions are more valuable than automated ones focused on low-hanging fruit.

Play #30

Experiment tracking

Innovation requires experimentation, however there is overhead to managing experiments effectively.

Being disciplined in the execution and evaluation of experiments is crucial to an organization. This play will walk you through how to effectively track experiments through the introduction of innovation tokens.

Use this play to:

- Establish a process to manage experiments
- Introduce innovation tokens
- Help teams experiment through hypothesis and answer "**can X?**", rather than "**how can X?**"

Expert tip

A tech radar is a great tool to track experiments. **The assess ring is your experiment zone**. Moving to the trial ring or hold ring is the clear path beyond the experiment and clearly indicates the decision.

Time required

60 minutes

Who participates

Architects

How often

As needed

“Let’s say every company gets about three innovation tokens. You can spend these however you want, but the supply is fixed for a long while… If you choose to write your website in NodeJS, you just spent one of your innovation tokens. If you choose to use MongoDB, you just spent one of your innovation tokens.”

Dan McKinley - Choose Boring Technology

Experiment tracking

1. Introduce the concept of innovation tokens. Use the quote on the previous page to get started.
2. Set experimentation boundaries by discussing and agreeing on the following criteria as a group.

The number of **innovation tokens (I)** should be...	The **maximum number** of simultaneous experiments **(M, M <= I)** we can sustain should be...	The **maximum default timebox (T, T <= 12)** for each experiment (in weeks) should be...	The **first experiments (N, N <= M)** we'll schedule should be...

If you've established a tech radar, the assess ring can help you out here.

3. Identify who should be involved in designing and running the experiments with the architects. Likely it will be Principal Developers and Engineering Manager type humans.
4. Start with an hypothesis, then design the first experiment from there. If you've allowed 3 innovation tokens per quarter, that's one experiment per month. Make sure you can design, run, and gather enough data in this time to make a call.

Hypothesis-driven development

We believe that <this capability>

Will result in <this outcome>

We will know we have succeeded when <we see a measurable signal>

Experiment tracking

5. Take it public. Publish the experimentation boundaries and the planned experiments for the quarter within the organization. Share them far and wide, make sure the development and delivery community is aware of the experiments. Information you want to record and share for each experiment includes:

Hypothesis	Experiment	Target metric	Actual result	Insights	Decision

Everything you record in this table contributes to measurable evidence and learning, make sure you're clearly articulating the experiment and a record exists to refer back to.

Make sure the decision is recorded and shared. The tech radar is a good tool to keep track of decisions.

"The key outcome of an experimental approach is measurable evidence and learning." - Barry O'Reilly

6. Set a regular cadence. Towards the end of each quarter, start planning the experiments for the following quarter so you can hit the ground running with experimentation in the new quarter.

7. There will be questions that arise along the way that can't be answered upfront, and answers will vary from team to team, domain to domain and organization to organization. Questions like, what if we've used all our tokens and need to do another experiment? What if a team-level experiment becomes a domain (or organization) level experiment? How do we adjust the number of innovation tokens after Q1? Make sure the group has open conversations and agrees on a way forward.

Play #32

Event storming

Event Storming is a workshop format for fast-paced domain modelling. The technique focuses on collaborative exploration by domain experts and the technical team. The format is flexible and can be used to explore the whole organization, a particular business process, or for tactical software design. It is applicable in a wide range of situations from greenfield initiatives to legacy migration needs.

When organizations move toward microservices, one of the main drivers is the hope for faster time to market. However, this aspiration only tends to be realized when services (and their supporting teams) are cleanly sliced along long-lived business domain boundaries. Otherwise meaningful features will require tight coordination between multiple teams and services, introducing friction in competing roadmap prioritization.

Use this play to:

- Create a domain model
- Learn about the domain as a group
- Identify domains and prioritize core domains
- Manage complexity; visualise essential complexity to avoid accidental complexity
- Align technology and the business

Expert tip

Getting the right people in the room is important - a blend of business and technical people who bring both the questions and the answers.

If running the exercise in-person, ensuring that you have enough wall space and stickies for modeling is the second key to success.

Time required

1 - 3 hours

Who participates

Architects, developers, business analysts, product folk, domain SMEs

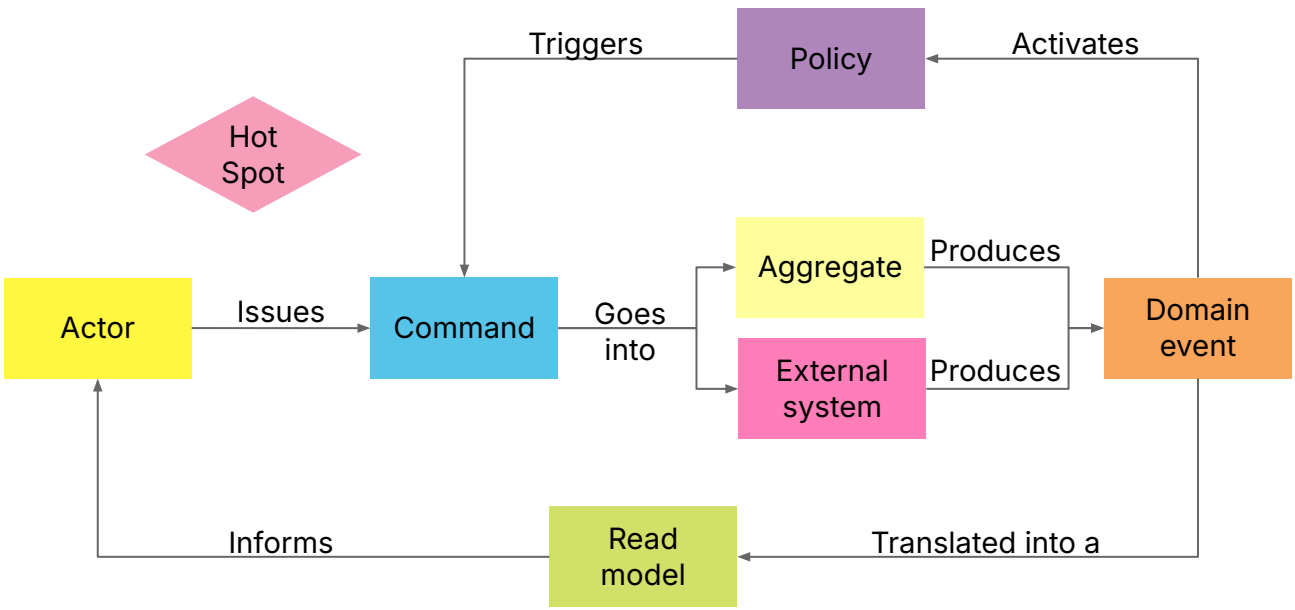
How often

As needed

Event storming

Actor	The persona/user/role who requests a command to be executed.
Command	Someone (user) tells the system to do something. The cause or trigger of an event.
External system	Systems that issue or receive commands within the domain. This includes third-party services or “external” services.
Aggregate	Logical group of domain objects (events, commands, actors, reactions) that can be treated as a single unit.
Domain event	Something that happened of significance in the business, something that is of interest to the domain expert.
Policy	A reaction to an event. Whenever X happens we do Y.
Read model (view)	Some actors may need additional information to determine their common choices, these are captured as read models.
Hot spot	Pain points or points of misunderstanding

Relationships between event storming concepts



Event storming

There are 3 main formats for running the workshop. Each format adds a new level and concepts, and drills down to specifics.

Big Picture Format Explore a business line that crosses corporate silos.	**Process Modeling Format** Model a business process.	**Software Design Format** The goal here is to create design artifacts for implementation.
Uses *event*, *actor*, *command* and *hot spot*.	Adds *system, policy, read model.*	Adds *aggregate*.

Start with the big picture format to identify problem areas and core domains. Use the group to prioritize and drill down using the process modelling format.

Choose concepts based on your needs. A lot of value and learning can be achieved even if only events are discussed.

Event storming

Start by creating the big picture format.

1	**Introduce event storming concepts**	Introduce the event storming workshop, covering: the purpose, why you are running this workshop and the desired outcome. Set expectations on how long it will take. Introduce the key concepts and let the group know which concepts you'll be capturing in the session, i.e. are you focussing on the big picture concept, or getting into the details with the process modelling format?
2	**Start with events**	Storm out the business process by identifying domain events and putting them on orange sticky notes. Domain events should be a noun + verb in past tense. For example, "ride ordered", "car arrived", "payment received". When you start with events, it forces the group to think about the behavior of the system. As opposed to thinking about the structure of the system. To get started try asking what the most important event is in the domain. Then, what events happen before that event and after that event. Place the events in order of occurrence (on a timeline). Capture pain points and points of misunderstanding related to the events as hot spots on pink sticky notes. Encourage participants to think out loud to keep the conversation going.

Event storming

Creating the big picture format continued...

3	**Record the commands that cause events**	Next, have the group start to record commands that cause the events. Capture commands on blue sticky notes. Commands should be stated in the imperative. For example, “order a ride”, “buy ticket”. A command might look like a reflection of the event (this is completely fine) It might be obvious and evident to those who are close to it. But to others it might not be. It’s important to capture the commands. There might be events that do not have an explicit command that causes it and that’s OK. Capture any extra context related to the command that comes up in conversation, for example scenarios and constraints. Capture pain points and points of misunderstanding related to the command as hot spots on pink sticky notes. Continue to encourage participants to think out loud to keep the conversation going.
4	**Record the actors that trigger commands**	Record the actors. An actor is the persona/user/role who requests a command to be executed. For example, “passenger”, “driver”. Add arrows to show the direction the events flow.

Event storming

If required, drill down to create the process modeling format. Finish the picture by adding bounded contexts (aggregate) to achieve the software design format.

5	**Record external systems**	Capture systems which the commands are issued to. Steer away from being bogged down by internal vs external system too early. Continue to add arrows as you go.
6	**Record policies**	Policies are rules that react to an event and trigger a command. Capture them on purple sticky notes in the format: “Whenever <Event>, then <Command>”. For example, “Whenever ride ordered, then send email”.
7	**Record read models**	Capture the information that actors need to make a decision on green sticky notes.
8	**Identify bounded contexts**	Draw boundaries (aggregates) around areas where definitions are clear within that context. Usually, boundaries exist where there are conflicting definitions or a definition doesn’t belong to a core function of the domain.

Play #33

Domain driven design

Domain driven design (DDD) is an approach to designing and developing software that is modelled based on the underlying domain. It is focused on building software that represents the business context, separating it from the technical concerns such as the persistence of data.

Domain modelling begins with strategic design, which develops a shared understanding of the high level business context, between business and technical stakeholders. It models business domains as Bounded Contexts, that are related to each other in Context Mappings. Within a bounded context, an Ubiquitous Language is established, that is shared and understood by all stakeholders.

The output of the strategic design is then partnered with the tactical design to model the construction of individual bounded contexts, in a manner that is consistent with the high level domain model. The tactical design consists of four building blocks; Entities, Value Objects, Aggregates & Domain Events

Use this play to:

- Establish a shared understanding of the business domain between domain experts and developers
- Identify team ownership boundaries
- Design software that matches the domain, leveraging the Ubiquitous Language within the Bounded Context
- Identify module boundaries

Expert tip

DDD helps us understand the business domain. Once modelled, it makes it easier for developers to distinguish *essential complexity,* inherent to the business, from *accidental complexity*, introduced by developers when designing.

Time required

2 hours - 5 days

Who participates

Architects, Technical Principals, Technical Leads, Domain SMEs/business representatives

How often

As needed

Domain driven design

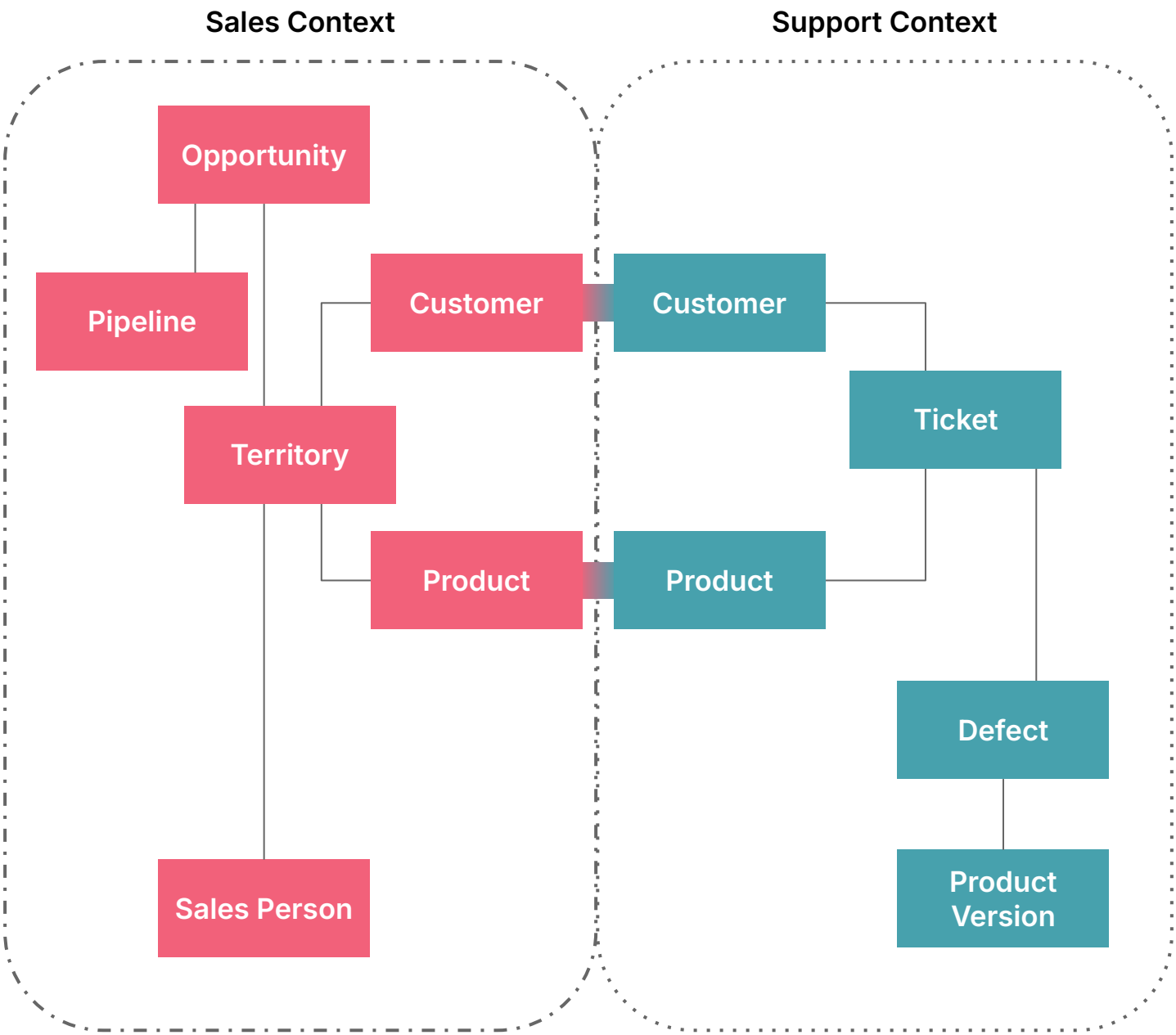
Bounded Context	A subdivision of the domain that defines its own independent domain model and Ubiquitous Language. It is aligned with specific business domains and ensures clear boundaries, preventing ambiguity between models.
Ubiquitous Language	A shared, precise language used by both developers and domain experts, reflecting the Domain Model and ensuring clarity in communication throughout the design and implementation of the software.
Context Mapping	An exercise for identifying and mapping relationships between different bounded contexts, showing how they communicate, influence each other, and manage dependencies, especially in terms of getting work done.
Aggregate	A cluster of Entities and optionally Value Objects, treated as a single unit responsible for keeping the state consistent, by placing transaction and consistency boundaries around them.
Aggregate Root	The central Entity in an Aggregate that serves as the only entry point for accessing the Aggregate. It is responsible for ensuring the consistency and integrity of the Aggregate as a whole.
Entity	An object that is uniquely identifiable and tracked by its identity, not by its attributes. Even as its properties or values change over time, its identity remains constant, distinguishing it from other objects. The identity gives an Entity continuity and makes it recognizable across different states or representations.

Domain driven design

Value Object	An object that represents a concept in the domain and is defined solely by its attributes. Value Objects are interchangeable when their attributes are identical, making them distinguishable from Entities.
Domain Event	A record of something significant that occurred in the domain, which is of interest to domain experts. Domain Events capture changes or actions and can trigger other processes or notifications in the system.
Domain Service	A stateless service that encapsulates domain logic, which doesn't naturally fit within an Entity or a Value Object. It typically represents a key domain operation or process that involves multiple objects but isn't tied directly to a specific Entity. Domain Services are part of the Ubiquitous Language and focus on the core business logic.

Domain driven design

Context Map representing relationships between Bounded Contexts



Map source.

Domain driven design

Start by modelling the strategic design

1	**Introduce Domain Driven Design**	Begin by introducing Domain-Driven Design and the purpose of the domain modeling workshop. Explain why it's being held, outline the desired outcomes, and establish expectations for the workshop duration. Introduce key DDD concepts, specifying whether the session will focus on high-level strategic design or detailed tactical design.
2	**Identify elements**	Follow Play #32 - Event Storming to identify the domains, events, bounded contexts and aggregates.
3	**Identify core, generic and supporting domains**	Follow Play #19 - Wardley Mapping to identify the core, generic & supporting domains.

Then, model the tactical design for each core domain

4	**Identify the entities and aggregate root**	Using the aggregates and domain events identified during event storming for the core domain, work with the domain experts to identify the entities within each aggregate. Once all the entities are defined, decide on the root entity, which will become the aggregate root. An aggregate has a single root entity, providing an identifier that will be exclusively used to lookup the entity.
5	**Identify the value objects**	Identify objects which exist within the aggregate, that do not have an identity and are defined by the values of their attributes, these are value objects. Value objects can be used as attributes of entities or other value objects and should be treated as immutable.

Play #34

Tech radar

Many enterprises suffer from long periods of unconstrained proliferation of technology choices. This pain is felt by the unwieldy technical landscape an engineer needs to grasp to become productive. More subtle issues arise when there is no visible criteria to help choose which technology to apply to problem "X".

An internal technology radar visualises the organization's posture towards the technologies currently used. Building a radar will force conversations around the current and future suitability of each technology. Keeping the radar up-to-date and visible provides an invaluable centrepiece for conversations with architects and delivery teams.

Use this play to:

- List all technologies currently used
- List sensible defaults (aka Adopt)
- Identify / agree current technologies being invested for suitability (aka Trial)
- Identify possibilities for future investigation (aka Assess)
- Explicitly list technologies not to be used,or ones to be decommissioned (aka Hold)

Expert tip

Consider a WIP limit on items in the Trial ring. Read into the use of Innovation Tokens as described by Dan McKinley here.

Do write details behind each blip. Include links to implementation details (Adopt), or specific context around applicability (Trial and Assess).

Localise quadrants and rings if needed: don't dogmatically stick with the original visualisation.

Time required

2 - 7 days

Who participates

Tech community

How often

Initial establishment, then revisit quarterly

Example

Tech radar

Thoughtworks has a BYOR (Build Your Own Radar) web tool. Find it here.

Play #35

Decision making framework

As noted in Play #29 - Fitness Functions, being an Architect is all about making significant technical decisions. As humans though, we're generally not great at making decisions. A decision making framework helps us to become better decision makers, removing emotion and bias, and weighing up the decision not just on expected outcomes (which is usually the default), but considering risk and other characteristics important to the organization.

Decisions and requirements don't need to - and shouldn't - be set up front. This is when the least information is known and therefore is the worst time to make decisions. The role of the architect is to minimize the number of decisions that are irreversible.[1]

Use this play to:

- Create a decision making framework for your organization
- Remove emotion and bias from the decision making process
- Make better decisions

Expert tip

Deciding not to decide is a decision. Decision should be made just-in-time and left to the last responsible moment. Decisions made too early lead to rework, increased complexity and higher risk.

Time required

45 - 75 minutes

Who participates

Architects

How often

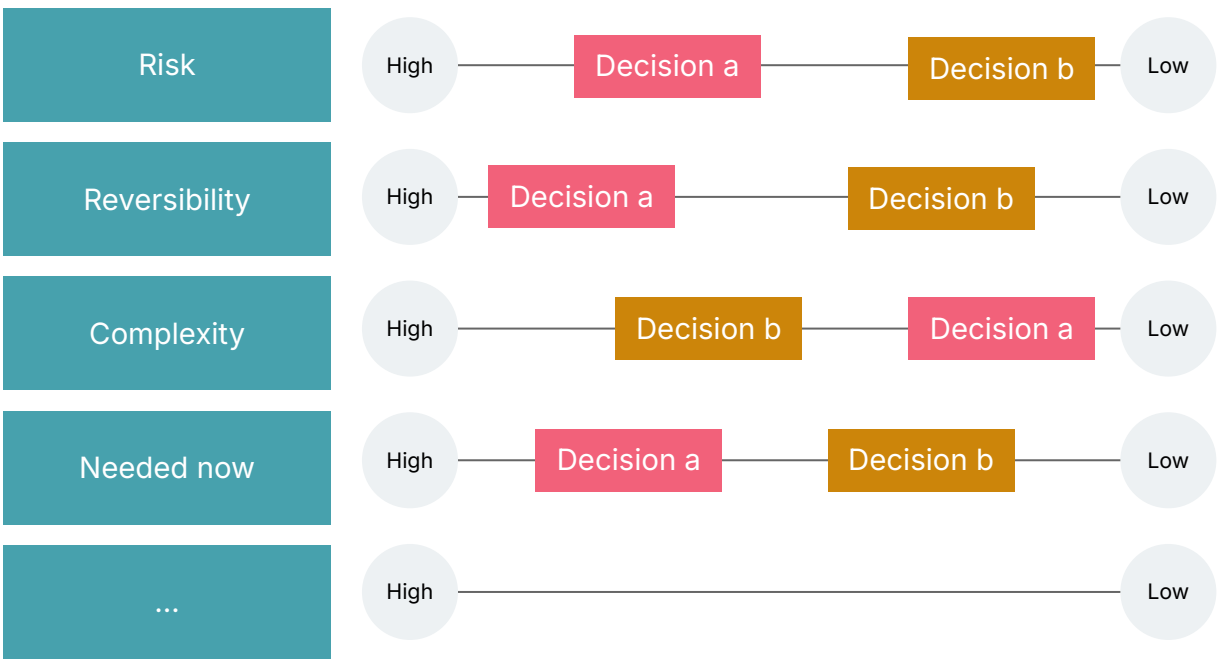
As needed

Decision making framework

1. **Define a set of characteristics** that relate to the organization or the domain in which the decisions need to be made. Start with risk and reversibility and add 3 - 5 more characteristics. Revisit your architecture principles and strategic goals to help define the characteristics.

Risk	What level of risk is involved in the decision?
Reversibility	How reversible is the decision?
Complexity	Will the decision introduce complexity?
Needed now	Is the decision needed now or can it be deferred?

2. **Assess your decision against each characteristic** from low to high. In some cases, low is what you're looking for (complexity) and in others, high indicates a better decision (reversibility).



The example above indicates *decision a* is more favorable than *decision b*. However, is the risk associated with *decision a* acceptable? This is something to trade off in the decision making process.

Play #37

Architecture principles

Architecture Principles define the underlying general rules and guidelines for the use and deployment of all IT resources and assets across the enterprise. They reflect a level of consensus among the various elements of the enterprise, and form the basis for making future IT decisions.

Each Architecture Principle should be clearly related back to the business vision and key architecture drivers.

Architecture Principles should be high level, meaning they shouldn't need to change frequently.

Establishing these principles and being deliberate in how they align with business strategy is an important exercise: it helps IT practitioners think about and understand the purpose of their work; and it allows those coming from a business perspective to appreciate the motivation and impact of technology decisions.

Use this play to:

- Define statements of purpose for how technology should be used across a whole enterprise and/or within specific divisions
- Guide decision making in development teams
- Inform people across the broader organization

Expert tip

Principles should be; aspirational, inspirational and (hold the Architects) accountable.

Principles need to be connected to outcomes.

Principles should be reassessed every 12 months.

Time required

2 hours

Who participates

Architects

How often

Initial establishment, then revisit annually

NOTE: The play format is based on Sebastian Mueller's The Design Principles Workshop and Yesenia Perez-Cruz's book, Expressive Design Systems and Workshop design principles.

Architecture principles

1	**Set the scene**	Spend 5 minutes talking through why there is a need for architecture principles and what makes a good principle.
2	**Get inspired**	Take 10 minutes to look at some principles from well established organizations to get inspired. Discuss what resonates and things people like or dislike.
3	**Review the business vision and strategy**	Architecture principles should be connected to the business vision and strategy - they exist not only to define the underlying general rules and guidelines for the use and deployment of all IT resources and assets across the enterprise, and reflect a level of consensus among the various elements of the enterprise, and form the basis for making future IT decisions. They also exist to enable technology to support the business in realising their vision through execution of the business strategy. Take time to create a shared understanding of the business vision and strategy.
4	**What principles exist today?**	Do any principles exist today? If so, share and discuss them. What we people like/dislike about the principles that exist today? Why are they working/not working?
5	**Brainstorm principles**	Brainstorm ideas. They don't need to be elaborate, one word is fine. The point is to get lots of ideas captured. Sometimes it is easier to define what something isn't. Let participants know they can brainstorm what the principles...

Architecture principles

5	**Brainstorm principles (continued)**	might be, along with what they aren't. Once you're done brainstorming, cluster like ideas and invite discussion. When you've established a shared understanding, vote on the ideas that resonate the most.
6	**Elaborate**	Take the top 5 - 7 ideas and start to elaborate on the details of each principle. Consider the implication of each principle on your tech team members (developers, tech leads, architects) and on your technology resources and assets. Ask participants to write one "as a tech team member..." and one "our technology resources and assets..." statement for each principle. Give people time to read through each statement, then ask them to vote on the statement that resonates most.
7	**Refine**	Refine your principles into succinct statements.
8	**Share**	Open your principles up for discussion and invite feedback. Make any necessary changes to ensure each principle is clearly articulated and can be understood by all in the org. Principles should be shared widely, including in a poster format. Each principle should have a title but can also carry a deeper explanation and examples of practices and tools that may help people making decisions.

Example

Architecture principles

Principles come in all shapes and sizes. A quick search online will return you hundreds of examples of Architecture Principles organizations have made public. It's important to ensure principles are designed by the organization, and the organization believes in them. An example from ex-Thoughtworker Evan Bottcher's work at Thoughtworks client, REA Group is included below.

Strategic goals

- **Enable scalable business**
  - More customers / transactions
  - Self-service for customers
- **Support entry into new markets**
  - Flexible operational processes
  - New products and operational processes
- **Support innovation in existing markets**
  - Flexible operational processes
  - New products and operational processes

Architectural principles

- **Reduce inertia**
  - Make choices that favour rapid feedback and change, with reduced dependencies across teams
- **Eliminate accidental complexity**
  - Aggressively retire and replace unnecessarily complex processes, systems, and integrations so that we can focus on the essential complexity
- **Consistent interfaces and data flows**
  - Eliminate duplication of data and create clear systems of record, with consistent integration interfaces
- **No silver bullets**
  - Off the shelf solutions deliver early value but create inertia and accidental complexity

Design and delivery practices

- **Standard REST / HTTP**
- **Encapsulate legacy**
- **Eliminate integration databases**
- **Consolidated and cleanse data**
- **Published integration model**
- **Small independent services**
- **Continuous deployment**
- **Minimal customisation of COTS / SAAS**

Play #38

Architectural diagrams and styles

Visualizing technical concepts is a powerful and ubiquitous way to convey complex ideas in a digestible and language-independent manner. Diagrams can efficiently communicate information when used correctly. However, the proliferation of diagramming techniques and tools has created a bewildering array of options for engineers and architects to select the "best way" to convey information visually.

This problem is compounded in Enterprise Architecture, where the scope of information is broader and deeper, often involving specific frameworks with bespoke tools and idiosyncrasies. These add significant design and maintenance constraints, leading to fragmented and ambiguous practices.

What should be a powerful tool for conveying complex information can easily become a "lawless territory" requiring unnecessary diagrammatic governance.

Use this play to:

- Identify indicators and symptoms of poor diagramming health
- Establish diagrammatic ways of working
- Create an Enterprise Diagram Atlas
- Set your organization up for success with clear, consistent, readable diagrams

Expert tip

Cross-team alignment on the “ways of working” for Enterprise Architecture diagramming as early as possible is a simple yet effective means of avoiding future ambiguity and unnecessary governance.

Time required

2 - 3 hours (initially)

Who participates

Architects, engineers, technical business analysts

How often

Initial session followed by 6-monthly review

Architectural diagrams and style

Observable indicators and symptoms of poor diagramming health in an enterprise are easier to spot initially. However, the longer you remain on the team, the harder they become to recognize as established practices are often accepted as "the way it is." The themes below highlight some common challenges encountered.

1. Visual clarity is in the eye of the beholder

Diagrams become overly complex and unreadable due to too much information, incorrect element sizing, poor use of color, poor contrast, inadequate labeling, lack of white space, missing objects, relationships, and flows. Consider diverse viewing conditions/devices.

2. Stylistic inconsistency

Visual inconsistency causes friction and makes interpretation difficult. The human brain prefers consistency in visual design.

3. Language barriers

Fragmented terminology across the enterprise and technology industries complicates understanding. Inconsistent terminology causes friction without a supporting glossary and common vocabulary.

4. Altitude sickness

Finding the right level of detail for a diagram is crucial. Too much or too little detail can alienate the audience and make the diagram irrelevant.

5. Perfect is the enemy of the good

Architecture diagramming exists solely to convey complex information in the most efficient way possible. Any superfluous and unnecessary and continuous “tweaking” is simply waste.

6. Agile software ≠ waterfall diagrams

Diagrams created in isolation and without continuous collaboration will simply cause unnecessary delays as a result of inevitable rework.

Architectural diagrams and styles

Based on the common challenges inherent to the majority of organizations, we can draft some fundamental rules to help set the foundation for alignment for all teams responsible for diagramming. An example would be similar to below.

1. Ubiquitous diagram language

The creation and agile maintenance of a **shared diagram glossary** across teams helps to align the language challenges by both the diagram authors and the reviewers.

3. It's all about perspective

In order to guarantee relevance of the diagram being created, it's vital to ensure the artifact is actually fit-for-purpose for intended recipient. The stakeholder in question may operate in a completely different domain. Concepts of "views" and "viewpoints' may be helpful here.

4. It's good to talk

Fail-fast, iterative collaboration must accompany diagram creation, review and evolution. Agree up-front on lightweight ways of working for diagram reviews. Consider review stages at 30%, 60% and 90% to provide early feedback.

4. Provide lightweight guidance for visual clarity

Instead of rigid style guides, offer example diagrams, reusable templates, and reference patterns that integrate naturally with a shared diagram language. This encourages consistency with minimal friction, supports collaboration and ensures diagrams remain clear and aligned.

5. Diagrammatic Prime Directive

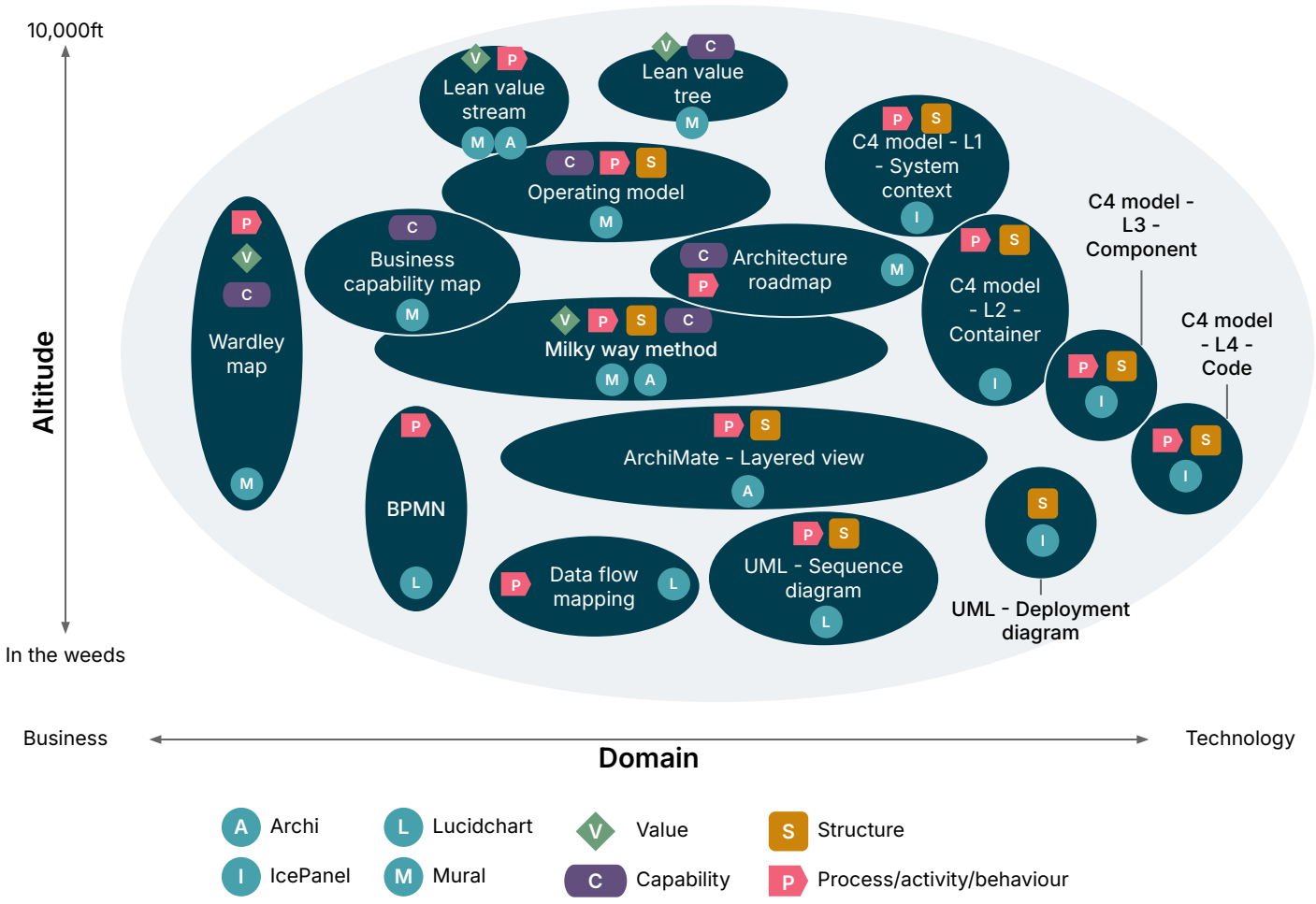
To prevent unnecessary emotional judgment of individual diagramming skills, apply a Prime Directive approach, fostering constructive feedback and a shared sense of improvement.

Architectural diagrams and styles

The Enterprise Diagram Atlas (EDA) is a technique that helps align everyone on the types of diagrams used across the organization and the areas of the enterprise they illustrate. In addition to covering business, technology, and altitude, the EDA can highlight the type of diagram (e.g., value, capability, process, structure) and the indicative tools to use.

Creating the EDA can be done through a simple workshop, either remotely using a tool like Mural or in-person with post-its, provided there is a digitally accessible version for everyone to explore.

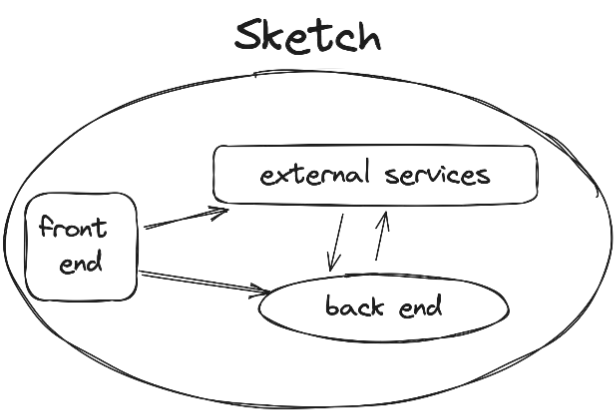
Enterprise Diagram Atlas

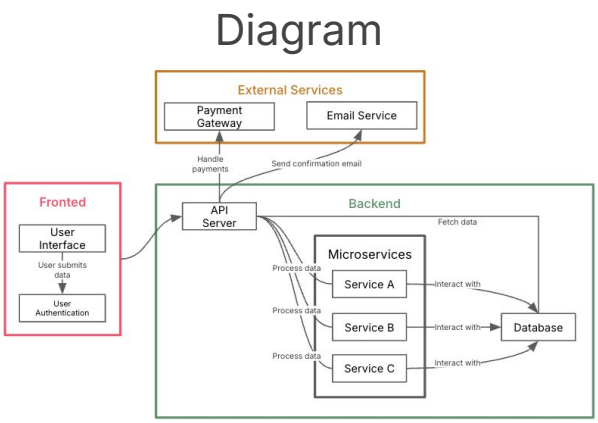


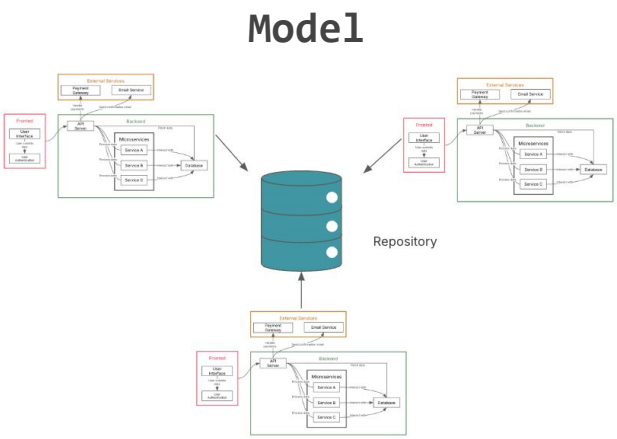
Architectural diagrams and styles

As illustrated by the EDA, there are literally **hundreds of diagram styles**, **Enterprise Architecture frameworks** and **methodologies** to choose from. The choice can be overwhelming so most people stick to a small(ish) number of reasonably well-known styles (i.e. boxes and arrows, C4). Unfortunately, an established enterprise architecture function will have (not necessarily realistic) expectations on “acceptable” diagram types and styles. These can include:

- A **sketch** - an informal, outline used for initial concept development.
- A **diagram** - a formal, medium to high-detail visualization used for detailed planning and communication.
- A **model** - a formal and comprehensive representation used for in-depth analysis and decision making.







Architectural diagrams and styles

In addition to expectations on “acceptable” diagram types, constraints imposed upon diagram editors as a result of the tooling and/or frameworks involved may add further challenges.

When an organization is heavily invested (commercially or emotionally) in a particular tool or framework it can result in significant “friction” as the **license constraints** of many Enterprise Architecture tools **prevent ubiquitous access** to engineering teams and at times, architects themselves.

To address these challenges and ensure all teams create consistent, clear, and readable diagrams, it is essential to reach an early consensus on the tools to be used. Consider tools like Lucidchart or Mural for the majority of diagrams across all teams. Additionally, ensure that everyone in the organization has access to the selected tools.

There will inevitably be exceptions, for example when there is a specific modelling requirement that requires more formal diagram creation for analysis purposes.

Steps to clear, consistent, readable diagrams

1. **Standardize diagramming tools:** Agree on a single platform (Lucidchart, Mural, etc.) to ensure consistent and accessible diagrams across all teams.
2. **Publish an Enterprise Diagram Atlas:** Define and describe diagram types and their uses throughout the organization
3. **Establish a diagram style guide:** Outline core principles, patterns, and styles to ensure readability and standardization across all teams.
4. **Establish diagrammatic ways of working:** Facilitate easy participation for contributors, reviewers, and users.
5. **Ensure accessibility:** Make sure all diagrams are discoverable and accessible to everyone in the organization.

Play #39

RFC (request for comment)

It is important for architectural options to be shared, asking for input from the delivery team, especially the developers. By sharing these options and allowing time and space for feedback and comments, teams can feel empowered to implement the solutions without distracting them from delivering value through features.

An RFC is a brief for each major work parcel (a Feature or Epic) or when the team encounters a 'departure' (something the team hasn't done before or where making a departure from an established architecture work pattern), which summarizes the research, spikes and findings which backed the technical options for the work parcel or departure. It is an easy way to collate research, document options and/or decisions which can be 'open' for peer review and input, and referenced in the future. An RFC can also be referred to as an Open Design Document or Open Design Proposal).

Use this play to:

- Improve the quality of your options with an open process for peer input
- Make better decisions
- Standardise documenting design decisions
- Create an archive that can be easily referenced
- Empower teams to contribute

Expert tip

RFCs can bring discipline and structure to the architecture function. They are a great tool to **help link design decisions to the architecture vision.**

RFCs are light-weight sharing mechanisms, which allow for feedback and comments.

Time required

60 minutes

Who participates

Architects, developers, tech community

How often

As needed

RFC (request for comment)

The intent of RFCs is to encourage open dialog, improve design outcomes, increase alignment, and reduce long-term rework.

To make this process effective it needs clear expectations of participation, and a well-documented process for recording and publishing decisions.

1. **Establish the process.** Design and agree on the RFC process. RFCs should be public and open for comment, a standard format and light-weight (i.e. you shouldn't be spending extensive time writing them). How will RFCs be shared within your organization? (i.e. chat channel, email list etc) How can people make comments? How and when will RFCs be reviewed?Introduce the concept of innovation tokens. Use the quote on the previous page to get started.
2. **Make it public.** Once you've established the process, make it known within your organization, especially within the technology function.
3. **Document**. Create a standard format (see example next page); document your first RFC.
4. **Share.** 'Open' the RFC for peer review and input. This broadens your options, ensures you don't miss trade offs or anything major and leads to better decision making.
5. **Run a review meeting.** Encourage teams to participate in a review session to discuss their feedback.
6. **Record the decision.** Record the decision; share the decision with interested parties.
7. **Review the decision.** Once the action decided on is completed; run a post implementation review. Was the decision the right one? What did you learn from this?

Note: When writing a RFC always use full sentences and avoid bullet points as much as possible. This will mean the RFC can serve a wider audience; people will be able to read and understand the proposal without the need for the author to be present.

Example

RFC (request for comment)

RFC | Title | Date	
Context	Draft 8 - 12 sentences based on your knowledge of the business and the technology landscape.
Terminology	Make it easy for people to read by including a list of keywords.
CFRs	What CFRs (cross functional requirements) are considered for this?
Solution	Include one or more solutions with rough, high-level sketches. A photo of a sketch on a whiteboard is fine. As is a screenshot of a sketch on an iPad. Consider the C4 Model for visualising software architecture.
RAID	Some questions to consider when thinking through risks, assumptions, issues and dependencies: • Are other teams dependent on you for this solution? • Are you dependent on other teams? • What technical risks are you aware of? • What are the risks/implications of delaying this decision?
Cost Analysis	Show the costing for the various tools/services that are to be used. Costs should be broken down by each tool/service as a monthly and annual cost. Highlight the running cost for the milestone annually (final annual cost for all tools and/or services combined).
Impact	What is the impact of this option on the business?
We have consciously decided to	List of decisions made that would seem counter to best practice, your vision, guidelines or principles.
By doing so we will accomplish the following	List of things you hope to advance.
Research and Spikes (summarized)	Summary of the research and spikes that support this design proposal with links to the more detailed research and spike documentation for more context.
ADRs	Architectural Decision Records that were created to support this work.

Play #40

Technical debt tracking

Technical debt tracking enables you to understand the short and long term impact of technical debt on upcoming features.

Software systems are prone to the build up of cruft - deficiencies in internal quality that make it harder than it would ideally be to modify and extend the system further. Technical Debt is a metaphor, coined by Ward Cunningham, that frames how to think about dealing with this cruft, thinking of it like a financial debt. The extra effort that it takes to add new features is the interest paid on the debt.[1]

Good management of this inevitable tech debt allows product owners to understand the whole product, including the cruft. Making the tech debt visible allows product owners to understand the complexity and effort required to deliver new features. Product owners are empowered to make the best decision as tech debt impacts the duration and effort required to deliver a milestone.

Use this play to:

- Bring tech debt to the forefront
- Help Product Owners understand the whole product and empower them to make decisions on tech debt
- Understand the short and long term impact of technical debt on upcoming features

Expert tip

Surface technical debt and bring it to the forefront of conversations. When Product Owners and Engineering Managers can see the whole product, including the debt, they are empowered to make better decisions. Don't let your organization fall into the trap of gradually increasing technical debt, creating rigidity and unnecessary risk.

Time required

5 - 60 minutes

Who participates

Technology organization

How often

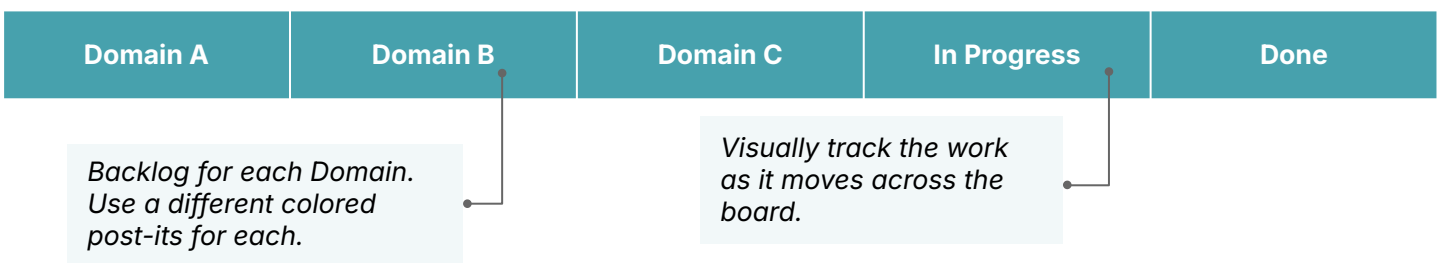
Fortnightly or monthly

Technical debt tracking

1. **Standardise the recording of technical debt.** Introduce a standard way to record technical debt that is simple, light-weight and open to anyone to record technical debt. Roll this standardisation out to the organization.

Example | Technical Debt Register	
Title	A simple title describing the debt.
Date, Author	Date added to Register, by whom.
Area of Concern	Component or Feature that this belongs to.
Severity/Impact	High / Medium / Low
Effort to address	High / Medium / Low
Consequences	What is the consequence of not addressing this.
Triggers	How will we know when is an appropriate time to address this? E.g. new feature development in this area; adding a third option to this scenario; talking to a different endpoint.
Injection point	Was this a deliberate decision or a discovered decision? For example, in looking through the code technical debt was discovered. It was clear that at the time we did not make conscious choices about it.
Possible Remedy	Any known remedies to reduce the debt.

2. **Make technical debt visible in the organization.** As Enterprise Architects it's important to make technical debt visible across the organization and to track the debt in an effective way so that the impacts and risks caused by the debt can be communicated not just within technology, but also to the business. Create a wall to visualise and track the debt. Decide on a regular cadence to review the wall with Product Managers, Engineering Managers and business representatives.



Play #41

Paying off technical debt

Technical debt, when left to pile up, slows the development process. Technical debt increases lead time to change and mean time to recover, leads to higher change failure rate and lowers deployment frequency.

In simple terms, technical debt is when technologies running in production require fixing, refactoring, modernizing, upgrading, reengineering, or replacing before or as part of implementing the strategic business requirements.

The significance of technical debt can vary. On one hand, it can be as simple as fixing a unit test. On the other hand, it may be as significant as reengineering complex monolithic applications.

Use this play to:

- Understand what technical debt means to technology, product and the business
- Start paying off technical debt
- Establish a regular forum to discuss and track technical debt

Expert tip

Continuously pay down technical debt, while understanding you never need to, or will, pay off all technical debt.

The more you develop, the more technical debt you'll accrue - it will just happen with the continued development of a system, the main thing is to keep technical debt from spiraling out of control.

Time required

2 - 3 hours

Who participates

Architecture team, product, tech and business representatives

How often

Monthly or quarterly

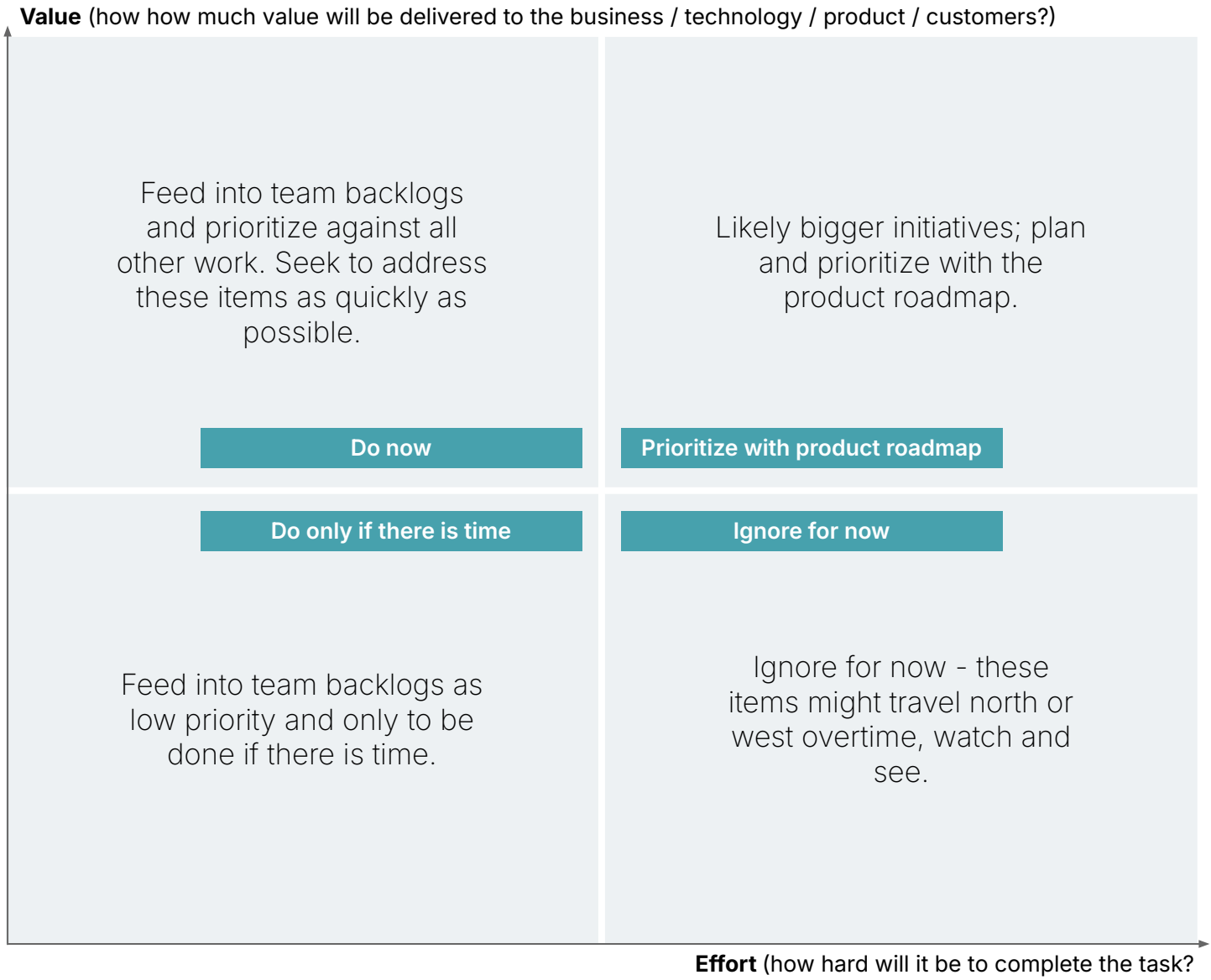
Paying off technical debt

1	**What does technical debt mean to the organization?**	Start by creating a shared understanding of what technical debt means to the organization, capturing the perspectives of the customer, the business, technology and product. As a group, answer the following questions: • What does technical debt mean? • What is the impact of technical debt? • What is acceptable technical debt? • What is unacceptable technical debt?
2	**Prioritize technical debt**	In Play #40 you've established a standardized way to record technical debt. You've visualized this debt somewhere and there is an awareness of it across the organization. Now it's time to prioritize the technical debt. Place technical debt cards in a value/effort quadrant to understand what should be done now and what can be ignored for now. See next page for example.
3	**Start paying off technical debt**	As a group of architects, product managers, engineering managers and technical leads, discuss how teams will work on technical debt. The 20% rule is a common way to ensure a sensible amount of technical debt is being addressed: reserve 20% of each team's capacity for technical improvements each iteration. On occasion, this may need to be a little higher, 20% is the minimum.
4	**Establish a regular cadence**	Establish a regular cadence to assess and prioritize technical debt. Organizations with high autonomy and agile maturity might only need bi-monthly or quarterly meetings, organizations at the other end of the scale may need more frequent touch points.

Example

Paying off technical debt

Visualize and prioritize technical debt using a value/effort quadrant. As you discuss each item of technical debt be sure to cover the impact if the debt is not addressed and the value that will be delivered when the debt is paid down.



Play #42

Architecture workload management

As Architects, it can be easy to work siloed away in individual domains. We might be tracking work on post-its, on a digital wall, or on a notepad. Surfacing and tracking architectural work across the organization creates transparency, surfaces bottle necks, highlights efficiencies or inefficiencies and opens up dialogue.

Outputs from this play can be fed into Play #9 - Operating Model: The Architecture Group.

Use this play to:

- Understand where your work is coming from
- Build a central wall to record and track all architectural work
- Establish a regular forum to discuss and track architectural work
- Get a view on how much work is in progress at once and introduce WIP (work in progress) limits

Expert tip

Once you've established a central wall (physical or digital) for all architectural work, share it far and wide. Make sure the organization knows where to find it, why it exists and how to comment, contribute or provide feedback.

Time required

60 minutes

Who participates

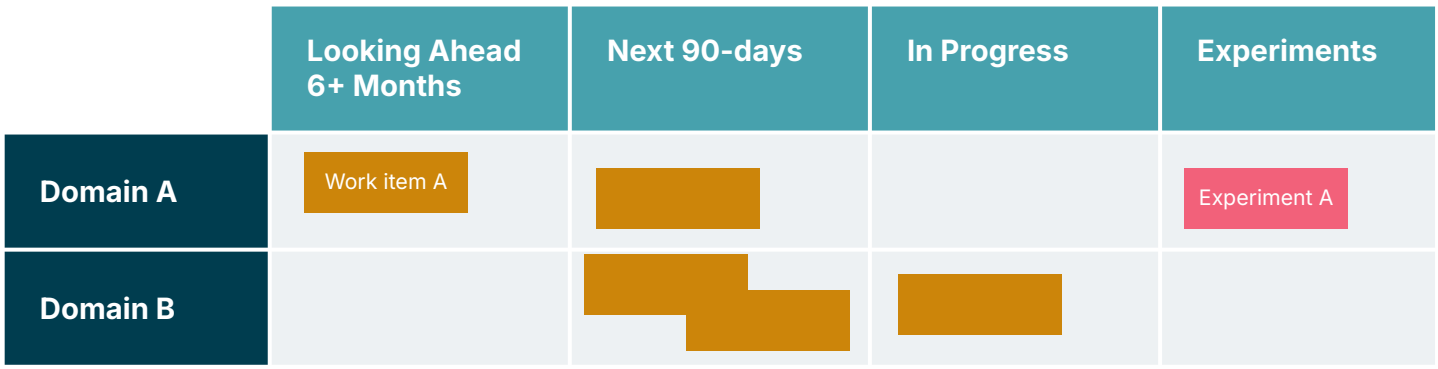
Architects

How often

Initial establishment, then weekly or fortnightly tracking

Architecture workload management

1. **Understand where the work is coming from.** As a group, spend a few minutes noting down where you work is coming from. Is it coming to you, or are you going out and finding it? How does the organization know what you're working on? How do you communicate WIP? Take time to reflect as a group to determine what the right mix is.
2. **Capture the work.** Set-up a board to record all known architecture work. Record the work at a high level initially, group by domain and timeframe. Track any known experiments in play too.



3. **Track the work.** Set-up a regular meeting to discuss and track the progress of your work. Record and track actions from the meeting; discuss experiments in play and the lessons learned; capture decisions made, keep a record of the artifact to stem from the decision.

Example Meeting Canvas

Actions	Experiments in Play	Decisions Made	Walk the Wall
Review open actions; record actions as they come up in the meeting.	*Discuss experiments in play. What has been learned? What actions need to be taken? Are there too many in play?*	*Record any decisions you've made; include a link to the relevant artifact.*	*Walk your wall of work.*

Play #44

Talent growth assessment

A talent growth assessment allows Architects to (a) get clear on core competencies they should be able to "do" as an architect and (b) reflect on their performance against these competencies and identify areas of strength and areas of growth both individually and as a collective.

Use this play to:

- Understand core competences as an Architect
- Identify individual areas of strength
- Identify individual areas of growth
- Understand the strengths and growth areas for a group of Architects

Expert tip

Before you share the self-assessment results with the group, do a safety check to ensure all participants are comfortable sharing and discussing their assessments.

Run the meetings without the Architect's manager present. Let the Architects know it's up to them to share their individual assessments with their manager and/or mentor.

Time required

2 - 3 hours

Who participates

Architects

How often

Every 6 - 12 months

Talent growth assessment

1	**Set the scene (offline)**	Depending on where the Architects are at, this exercise could be confronting. Give them a heads up and some time to reflect before you ask them to take a self-assessment. Email the Architects with your intention and give them a preview of the competencies (next page). Ask them to reflect on these competencies with respect to themselves ahead of the self-assessment meeting.
2	**Self-assessment (meeting 1)**	If you're pressed for time you could squeeze this into 45 minutes, however 60 minutes is recommended. Share the guides, competencies and a self-assessment sheet with each participant. Let them know it's best to go with their gut and not dwell on the level too long. Encourage them to read the competency description and not take the title at face value. Once you've set it up, there won't be a lot to do as facilitator. There may be an occasional question here and there, otherwise, you're creating time and space for participants to self-assess. Once finished, ask participants to email you their assessments. Let them know you will aggregate result and they will be shared and discussed in the next meeting.
3	**Aggregate results (offline)**	There are a number of ways you could cut the data (we've shown you one). Aggregate the results. Prepare and send individual spider graphs to each participant ahead of the next meeting. Also ask if they are comfortable having their assessment shared with the group.
4	**Share back results and discuss (meeting 2)**	Share back the aggregated results and open up a discussion with the group. What stands out? What surprises or confuses them? What areas would they like to focus on individually or collectively? How might they do so?

Toolkit

Talent growth assessment

The talent growth assessment is based on the Thoughtworks Systems Architect archetype.

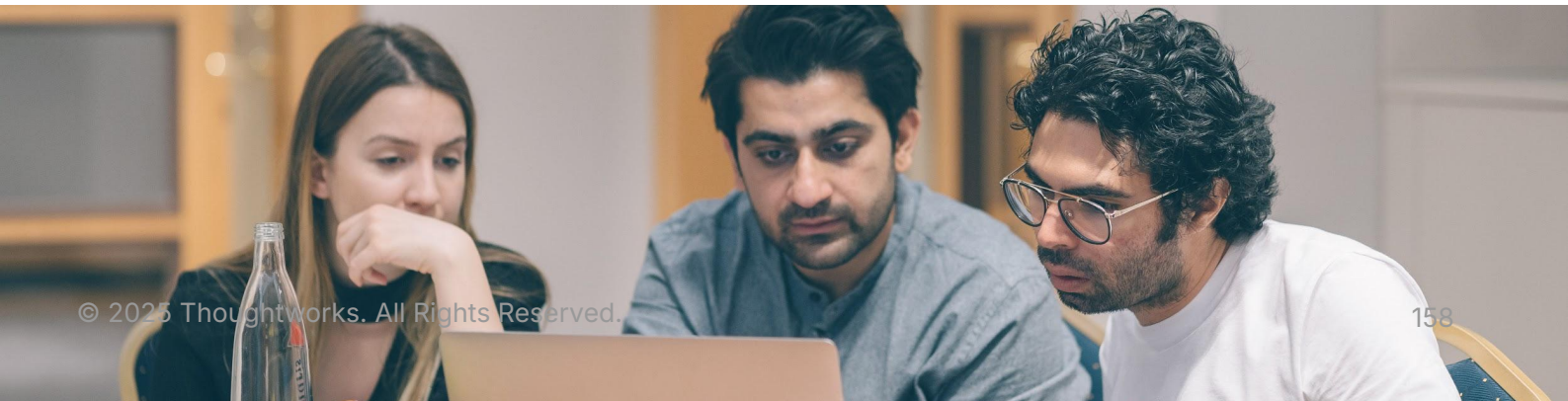
Thoughtworks describes the Systems Architect archetype as:

> **A holistic thinker who drives and evolves a complex technical vision with their deep knowledge of integration and architecture across an enterprise. Encourages flexible thinking and appreciation of new, emerging insights and multiple perspectives as a way to solve problems and steer technical direction. Able to straddle business and technical concerns, creates options and solutions that address business needs and longer-term strategic concerns.**

There are 18 competencies associated with this archetype:

- Domain modeling
- Governance, compliance and risk
- Influencing and persuading
- Lightweight architecture governance
- Modeling total cost of ownership
- Navigate organizational politics
- Risk Management
- API and service design
- Distributed systems architecture
- Drives to outcomes
- Effective communication
- Enterprise architecture
- Evolutionary architecture
- Integration
- Stakeholder management
- Systems thinking
- Technical decision making
- Technical visioning and roadmapping

Self-assessment sheet | Competency descriptions | Sample data aggregation



Go-to responses for common CTO/CIO questions

Common CTO/CIO questions and go-to responses

CTOs and CIOs face a constant stream of critical inquiries. This section provides clear and concise answers to the most frequently asked questions.

We've compiled a list of common questions, along with practical approaches and strategic plays to help you navigate these key conversations with confidence. By effectively aligning people, processes, and technology assets, you can optimize business outcomes and drive growth.

Each question is paired with a concise explanation and a set of actionable plays you can utilize to formulate a well-rounded response.

Common questions

Can you provide a 10,000-foot view of our system landscape, including key systems, their primary functions, and the responsible teams?

Go-to response

Begin by creating a holistic view of the system landscape. This involves combining elements of a high-level logical architecture with a capability map and a view of what teams own these.

Where possible, leverage existing artifacts like Event Storming or Domain-Driven Design models to inform analysis. If they don't exist, create them.

Plays to answer the question

15 Capability mapping

5 KYTE: Know Your Technology Estate

32 Event Storming

33 Domain driven design

23 Emergent capability mapping

38 Architectural diagrams and styles

How can I scale my architecture practice without increasing the number of architects?

Go-to response

Scale smart by empowering your architects to lead, not just do.

Ensuring software teams experience true autonomy raises a key problem: how might a small group of architects feed a significant number of hungry, value-stream-aligned teams? Why? Because in this environment architects now need to be in many, many more places at once, doing all that traditional "architecture".

What we need is a workable way to approach the human-scaling challenges of team autonomy and the architectures which manifest as a result.

Architecture need not be a monologue; delivered top-down from a centralized few. To scale your architecture practice without adding architects, empower your architects to become strategic leaders, not just individual designers.

This means they step away from team-level architecture decisions and move into a facilitator role. They focus on establishing principles and practices that guide and empower teams so that they can focus on systems-level architecture, turning knowledge stocks into knowledge flows, and harvesting and operationalizing the technical responses to the technology strategy.

Leverage the architecture advice process; a series of conversations, driven by a decentralized and empowering decision-making technique, and supported by four learning and alignment mechanisms: Decision Records, Advisory Forum, Team-sourced Principles, and a Technology Radar.

Plays to answer the question

11 Lightweight architecture decision records

37 Architecture principles

34 Tech radar

What is the role of an enterprise architect in a modern, evolutionary, cloud-native, you-build-it-you-run-it technology organization?

Go-to response

Enterprise Architects typically see themselves as gatekeepers and checkers so when a team of developers is encouraged to make their own decisions and operate with autonomy the EA is often left wondering why anyone would consult them at all. They also feel a loss of authority without a commensurate lessening of accountability. This has to be worked through. The role needs to shift to consultant and facilitator rather than gatekeeper. Instead of slowing change, the EAs need to see themselves as enablers of change. Instead of long solution documents and ponderous review boards, they can use lightweight architectural decision records and architectural principles. Instead of lists of rules, human review and excessive change control, they can use built-in guardrails and automated control at the point of change. Enterprise Architects need to shift from imposing top-down edicts to facilitating two-way communication up and down the hierarchy. They exert influence on the engineering teams on the ground and they provide transparent visibility of that change upward to executives.

In some cases, particularly within the regulated financial sector, we observe a preference for maintaining a clear distinction between the roles of architecture and engineering. Engineering is expected to move faster all the time whereas architecture is held accountable for maintaining order and preventing disasters. These are inherently opposing goals and require constant balancing. In those situations EAs need to be particularly collaborative and work with teams on the ground to help them understand the risks and how to mitigate them.

Plays to answer the question

1 Organizational context

9 Operating model: The architecture group

11 Lightweight architecture decision records

10 Enable effective decisions

35 Decision making framework

13 Responsibilities matrix (RACI)

12 Artifacts on a page

39 RFC (request for comment)

34 Tech radar

37 Architecture principles

28 Execution strategy

27 Path to production

36 Architecture in the open

31 Automated control at the point of change

How do I ensure consistency across my technology estate?

Go-to response

To achieve consistency across the technology estate, two key aspects must be addressed: defining **what** consistency means and **why** it matters, and effectively communicating organizational decisions.

1. Define consistency and its value
Start by defining what consistency means for the organization and why it matters. Document this clearly and ensure it's understood across teams. Highlight the value, such as reducing platform complexity or easing engineers' cognitive load. Also, recognize that flexibility can be valuable for innovation and new technology adoption. By defining where flexibility is appropriate, you can support innovation while maintaining overall consistency across the technology estate.

2. Effectively communicate organization decisions
Consistency thrives on clear, aligned communication. Develop a communication strategy to ensure all teams understand and adhere to the defined standards. This could include:

- Establishing **architecture principles** to set clear guidelines and standards across teams.
- Maintaining a **tech radar** to track technologies—highlighting which are sensible defaults, which are being trialed, and which are under consideration. This structured process allows teams to experiment while keeping visibility on emerging technologies.
- Standardizing **architectural diagrams** to ensure clarity and uniformity across the board.
- Documenting **architectural decisions** to provide transparency and a clear reference for future decisions, including the reasoning and context behind each decision.

Plays to answer the question

10 Enable effective decisions
34 Tech radar
9 Operating model: The architecture group
11 Lightweight architecture decision records
37 Architecture principles
38 Architectural diagrams and styles

How does our current technology infrastructure support our business goals, and what are its strengths, weaknesses, and opportunities?

Go-to response

To understand how technology is contributing to the business vision and goals, a comprehensive enterprise architecture assessment will provide a clear picture of the current state, including strengths, weaknesses, and opportunities related to the technology landscape.

The assessment will:

- Identify the robust and reliable aspects of the technology architecture that effectively support business operations
- Uncover areas where technology hinders progress or creates limitations
- Identify areas where technology can be leveraged to improve efficiency, agility, or innovation
- Highlight areas where technology is delivering efficiency improvements, and is enabling the business to achieve their goals and realize their vision.

A technology assessment provides a deep understanding of the current state - positioning you to create a strategic roadmap for technology investments that directly contribute to achieving the business vision and goals.

Plays to answer the question

4 Business strategy

1 Organizational context

6 Stakeholder mapping

2 Architecture context

15 Capability mapping

3 Data flow mapping

5 KYTE: Know Your Technology Estate

32 Event Storming

22 Systems health assessment

16 Capability assessment

18 Capability gap analysis

19 Wardley mapping

When businesses face challenges, how do you approach determining if there's an architectural root cause?

Go-to response

When being asked if a business challenge has an underlying cause tied to architecture, start by clarifying what is the scope of architecture and distinguishing it from engineering. Senior leaders may conflate the two, so ask clarifying questions to fully understand the specific concern before jumping to solutions.

To determine if a business challenge has an architectural root cause the first thing to do is to align with business outcomes. Start by clearly understanding the specific business challenge and its impact on the desired outcomes. Then assess if there is a clear link between the business outcome/s and architecture. This helps determine whether the issue stems from a misalignment between architecture and business goals.

Take a collaborative approach by involving business, technology, product, and delivery stakeholders to ensure everyone has a shared understanding.

Next, explore how architecture is supporting the business in achieving its goals, and analyze system architecture to identify weaknesses or limitations contributing to the challenge. Finally, assess if the right architecture capabilities and practices are in place, considering whether they're centralized or distributed and what needs to change to better support the desired outcomes.

Plays to answer the question

14 Operating model on a page

1 Organizational context

34 Tech radar

9 Operating model: The architecture group

How do I create a connection between the work of the EAs and the teams doing the 'actual' work that provides the right mix of freedom and constraint?

Go-to response

Creating a meaningful connection between the work of Enterprise Architects (EAs) and the engineering teams requires rethinking governance, processes, and how architecture can enable - not restrict - team autonomy. The goal is to strike the right balance between freedom for teams to innovate and the necessary guardrails that ensure alignment with overall business goals.

1. Examine governance and processes to identify where they hinder progress and where they can be made more lightweight, without compromising quality or security. Governance should empower teams. Shift the focus from rule enforcement to aligned autonomy.

2. Consider agile flight levels to create connections across different layers of the organization - strategic, coordination, and operational. This approach helps bridge the gap between high-level architectural thinking and day-to-day engineering work, ensuring alignment while still allowing flexibility for teams to make decisions at the operational level.

3. Align with actionable goals instead of setting rigid rules. Clear and relevant goals that teams can easily relate to and act upon will provide teams autonomy to choose how they achieve them, while the architecture vision and principles provide guidelines to follow.

4. Self-service automation tools and paved roads help remove friction. By offering these built-in paths, teams can stay within guardrails while maintaining their agility. Automation should be a cornerstone here, with guardrails embedded in CI/CD pipelines, security scans, and other processes that allow teams to innovate safely.

Plays to answer the question

26 Optimize outcomes: align business and technology

38 Architectural diagrams and styles

24 Business platform strategy

10 Enable effective decisions

28 Execution strategy

11 Lightweight architecture decision records

23 Emergent capability mapping

9 Operating model: The architecture group

39 RFC (request for comment)

To optimize our application landscape, which applications could benefit from consolidation or retirement?

Go-to response

Optimizing your application landscape begins with understanding which applications add value and which may be redundant. Capability mapping is a useful starting point, as it helps identify shared capabilities across applications. However, a more direct approach is to evaluate each application's relevance by considering the importance of the business processes it supports and its contribution to those processes. This should be done collaboratively with business, technology, product, and delivery stakeholders to ensure alignment.

One method is to set up a matrix that assesses the criticality of both the process and the applications supporting it. This allows for informed decisions on consolidation or retirement:

- **Strategic investment** - an application that is critical to a critical process
- **Wasted investment** - an application that is non-critical to a non-critical process
- **Middle ground** - applications that fall somewhere in between these two extremes require a deeper analysis. For example, a non-critical application supporting a vital process may still need to be maintained, but there could be opportunities to consolidate or migrate it to a more efficient platform.

Mapping out the relationship between processes and applications facilitates a more focused discussion around which applications to merge, consolidate, or retire. This approach ensures that the organization prioritizes strategic investments and reduces waste in maintaining unnecessary applications.

Plays to answer the question

8 Future, backwards

15 Capability mapping

16 Capability assessment

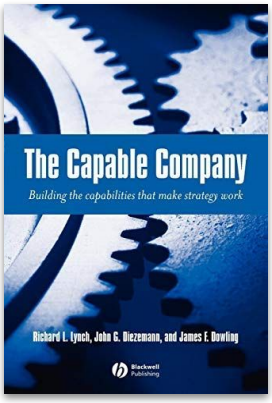
35 Decision making framework

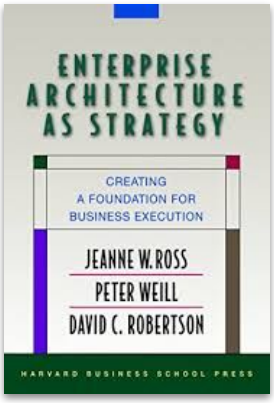
13 Responsibilities matrix (RACI)

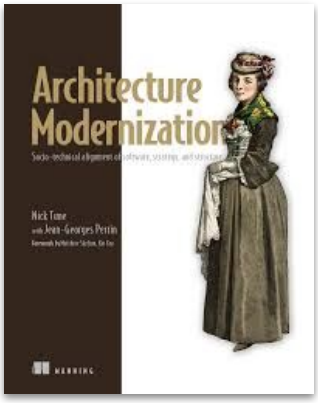
39 RFC (request for comment)

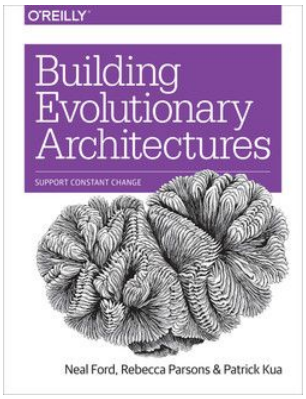
References and resources

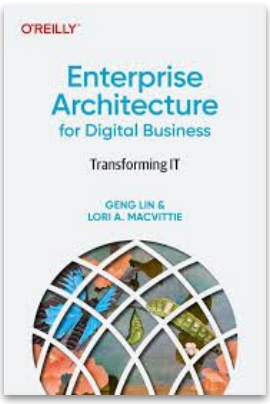
Recommended reading

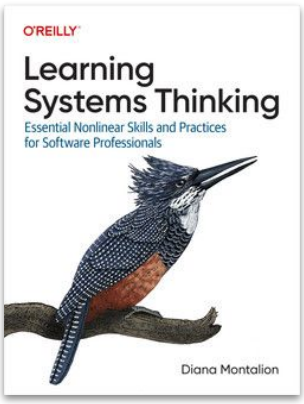


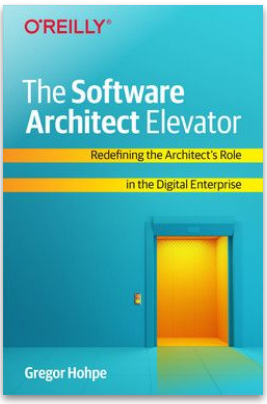


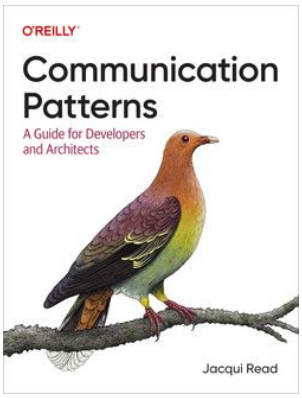


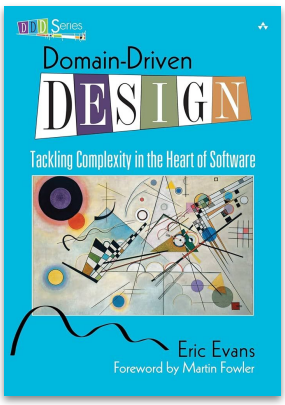


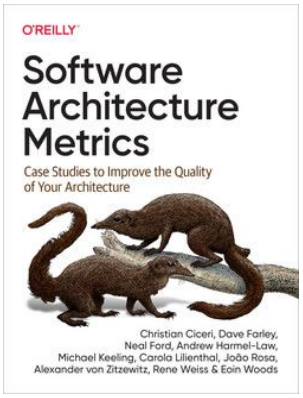












References

Bottcher, E. (2018, March 5). What I Talk About When I Talk About Platforms. Retrieved from https://martinfowler.com/articles/talk-about-platforms.html

Brown, S. (n.d.). The C4 model for visualising software architecture. Retrieved from https://c4model.com/

Conway, E. (1968). How Do Committees Invent? Retrieved from https://www.melconway.com/Home/Committees_Paper.html

Ford, N., Parsons, R., & Kua, P. (2017). Building Evolutionary Architectures. O'Reilly Media.

Fowler, M. (2014, January 15). Bounded Context. Retrieved from https://martinfowler.com/bliki/BoundedContext.html

Fowler, M. (2019, May 21). Technical Debt. Retrieved from https://martinfowler.com/bliki/TechnicalDebt.html

Harmel-Law, A. (2021, December 15). Scaling the Practice of Architecture, Conversationally. Retrieved from https://martinfowler.com/articles/scaling-architecture-conversationally.html

Hohpe, G. (2020). The Software Architect Elevator: Redefining the Architect's Role in the Digital Enterprise. O'Reilly Media.

Hunter, K. (2016). *Irresistible APIs: Designing Web APIs That Developers Will Love*. Manning Publications.

Jiwa, B. (2018). Story Driven: You don't need to compete when you know who you are. Perceptive Press.

Lombardo, C. T., McCarthy, B., Ryan, E., & Connors, M. (2017). Product Roadmaps Relaunched. O'Reilly Media.

Lynch, R. L., Diezemann, J. G., & Dowling, J. F. (2003). The Capable Company: Building the Capabilities That Make Strategy Work. Wiley-Blackwell UK.

McKinley, D. (2015, March 30). Choose Boring Technology. Retrieved from https://mcfunley.com/choose-boring-technology

Mueller, S. (2020, September 2). The Design Principles Workshop. Retrieved from https://uxdesign.cc/the-design-principles-workshop-2fe4e6637b35

Open Group Consortium. (n.d.). Architecture Framework. Retrieved from https://pubs.opengroup.org/architecture/archimate3-doc/ch-Example-Viewpoints.html

O'Reilly, B. (2014, October 18). How to Implement Hypothesis-Driven Development. Retrieved from https://www.thoughtworks.com/insights/articles/how-implement-hypothesis-driven-development

Perez-Cruz, Y. (2019). Expressive Design Systems. A Book Apart.

References

Picot, R. (n.d.). Design principles workshop. Retrieved from https://www.figma.com/community/file/1051212964426062558/design-principles-workshop

Portman, D. (2020, September 23). re you an Elite DevOps performer? Find out with the Four Keys Project. Retrieved from https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

Powers, S. (n.d.). Future Backwards: A game for determining next steps towards a desired outcome. Retrieved from https://www.adventureswithagile.com/future-backwards/

Pryce, N. (n.d.). ADR Tools. Retrieved from https://github.com/npryce/adr-tools

ThoughtWorks (n.d.). Build Your Own Radar. Retrieved from https://www.thoughtworks.com/radar/byor

ThoughtWorks. (n.d.). Enterprise Guide to Platform Thinking: What it Can Do for Your Business. Retrieved from https://www.thoughtworks.com/en-au/insights/e-books/enterprise-guide-to-platform-thinking

ThoughtWorks. (n.d.). Prime Directive: Not Just Star Trek. Retrieved from https://www.thoughtworks.com/en-gb/insights/blog/prime-directive-not-just-star-trek

Zobrando, M. (2013, November 11). Introducing Event Storming. Retrieved from https://ziobrando.blogspot.com/2013/11/introducing-event-storming.html

Thoughtworks lightweight enterprise architecture

Thoughtworks' Lightweight Enterprise Architecture accelerates your digital transformation by aligning business and technology, reducing time-to-market, and improving cost-efficiency. Our approach features built-in handover points to modern team structures, ensuring a smooth transition to agile delivery. Contact us today to learn how we can help you achieve tangible results.

Thoughtworks is a global technology consultancy that integrates strategy, design and engineering to drive digital innovation. We are 10,000+ people strong across 48 offices in 17 countries. Over the last 25+ years, we've delivered extraordinary impact together with our clients by helping them solve complex business problems with technology as the differentiator.

/thoughtworks

Published: March 2025