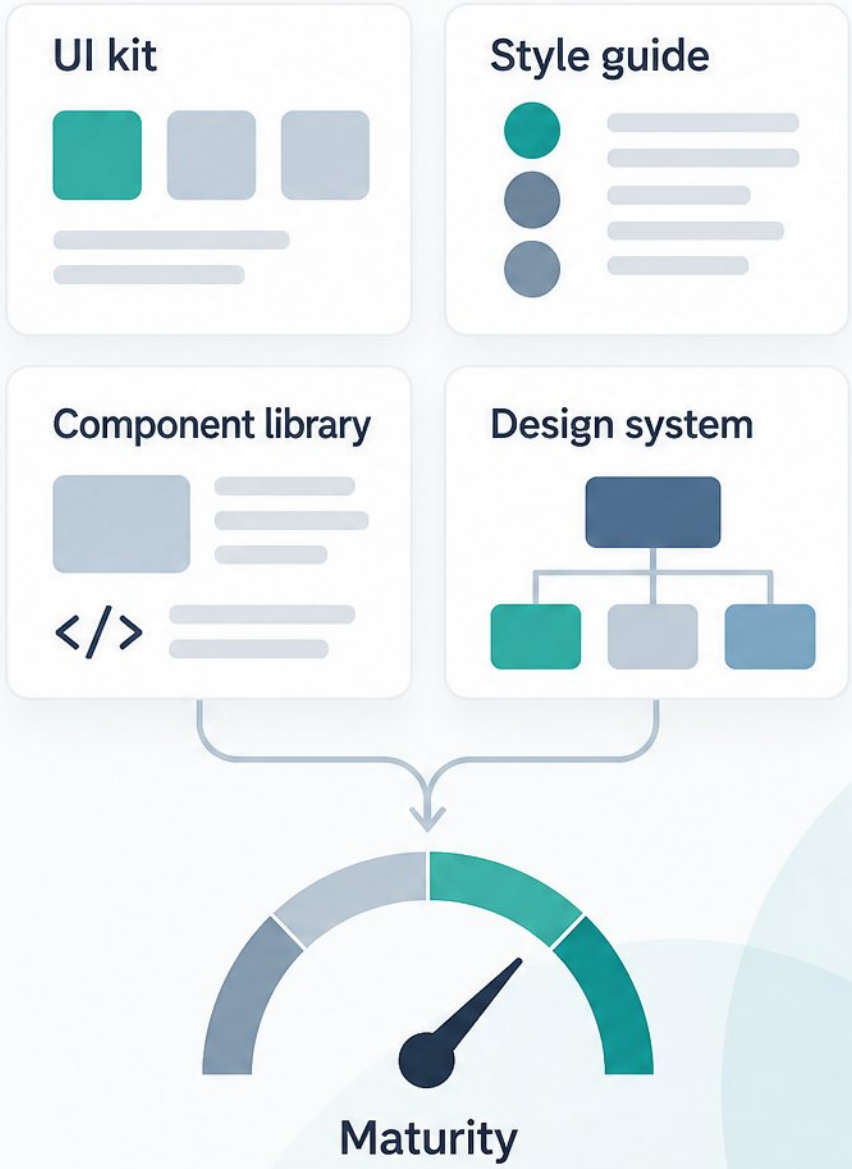
Types of Design Systems: An Overview

- System type sets governance, tooling, and daily design-engineering workflow
- The spectrum runs from a single shared UI kit to a multi-brand system-of-systems
- Core decision criteria: organizational scale, product diversity, governance, platform coverage
- Scope: define each type, diagnose your current model, choose a target
- Terminology is contested across the industry — align on working definitions early



Why 'Type' Beats 'Maturity' as a Starting Lens

- Four artifacts get confused: UI kit, style guide, component library, design system
- Maturity models describe health; type describes architecture and operating model
- High maturity at a library type is real — and common
- Type is a fit question, not a sophistication score
- Apply both lenses: type to choose, maturity to improve

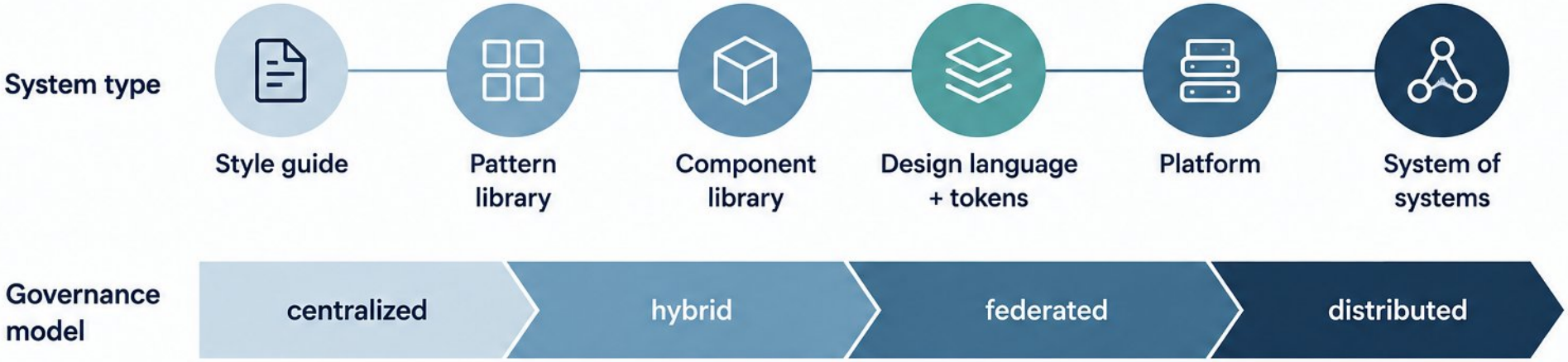


From Style Guide to Design Platform: A Short History



- Static PDF brand books → CSS variables → token pipelines and platforms
- Salesforce arc: 2013 pattern library and style guide; “design tokens” named for cross-platform values
- Early 2015: Landmark Design System, built on HTML blueprints, CSS framework, and tokens
- SLDS follows in 2015; early 2016 CSS framework joins core Salesforce code
- 2015–2016: SLDS, Material Design, Polaris, Carbon become versioned products with release notes
- Artifact shifts each stage: documents → code packages → APIs → services
- Many organizations run several stages at once; that is normal, not failure

The Type Spectrum at a Glance



- Working set: style guide, pattern library, component library, design language + tokens, platform, system of systems
- Governance maps to type: centralized, hybrid, federated, distributed
- Separate kind (token, component, pattern, template) from domain (forms, navigation, overlays)
- Complexity is not a classifier — a simple system can be mature
- Reality check: about half of teams run centralized; hybrid and federated cover most of the rest
- Each type gets its own diagnostic question in the sections ahead

Style Guides and Brand Systems

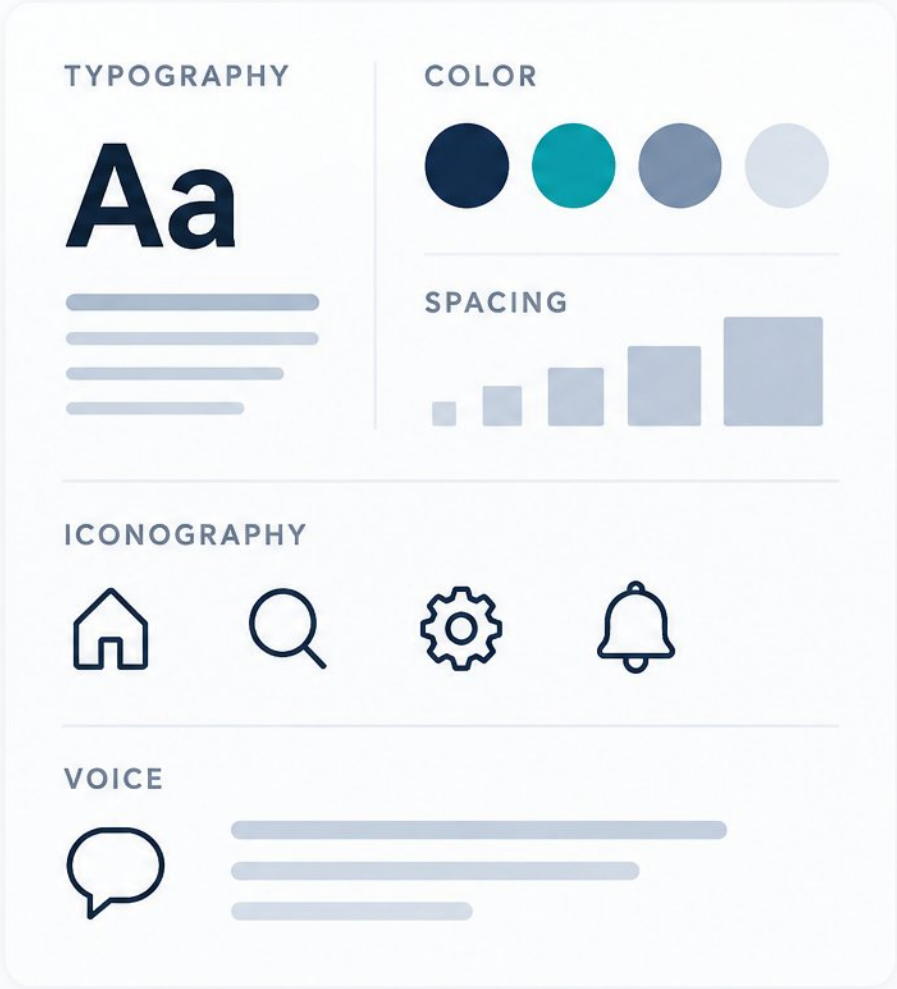
Scope: brand, typography, color, spacing, iconography, and voice

Right type when the problem is expression, not code reuse

Failure mode: static guides drift and need human interpretation

Evolution: brand guide → design language with enforceable tokens

Diagnostic: can a new team ship on-brand without asking a designer?



Pattern Libraries vs. Component Libraries

Pattern Library

Guidance-first

Context

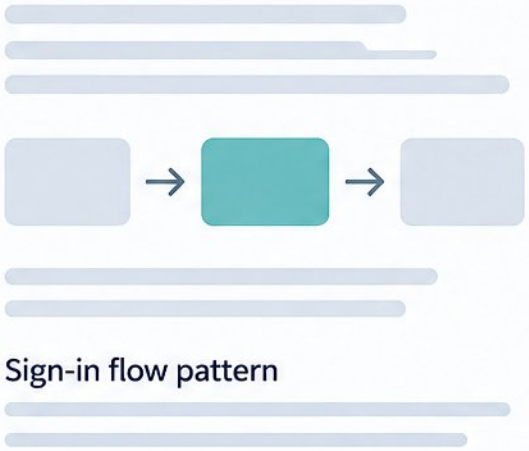
Problem

Solution

Composition

Rules

Anti-patterns



Component Library

Implementation-first

Props

States

Snippets



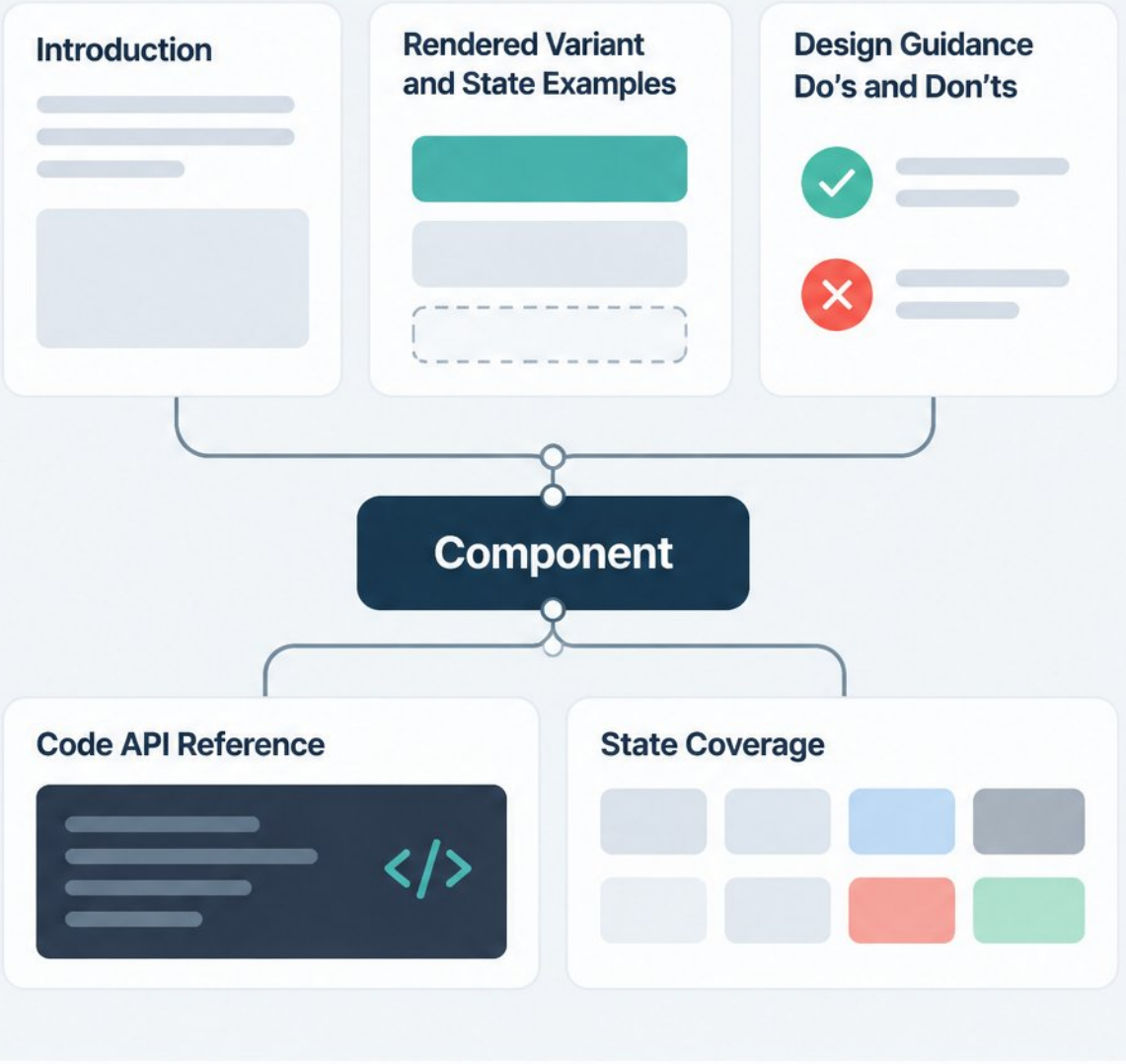

```
<Button
 variant="primary"
 size="medium"
 disabled={false}
>
  Label
</Button>
```

- Pattern library: guidance-first — how components compose into layouts and flows
- Component library: implementation-first — coded elements with props, states, and snippets
- Both are legitimate types; mature systems often run both
- Pattern pages cover context, problem, solution, composition, rules, anti-patterns
- Link out for component specifics; never restate component docs
- Failure modes: orphaned components, undocumented variants, design–code drift

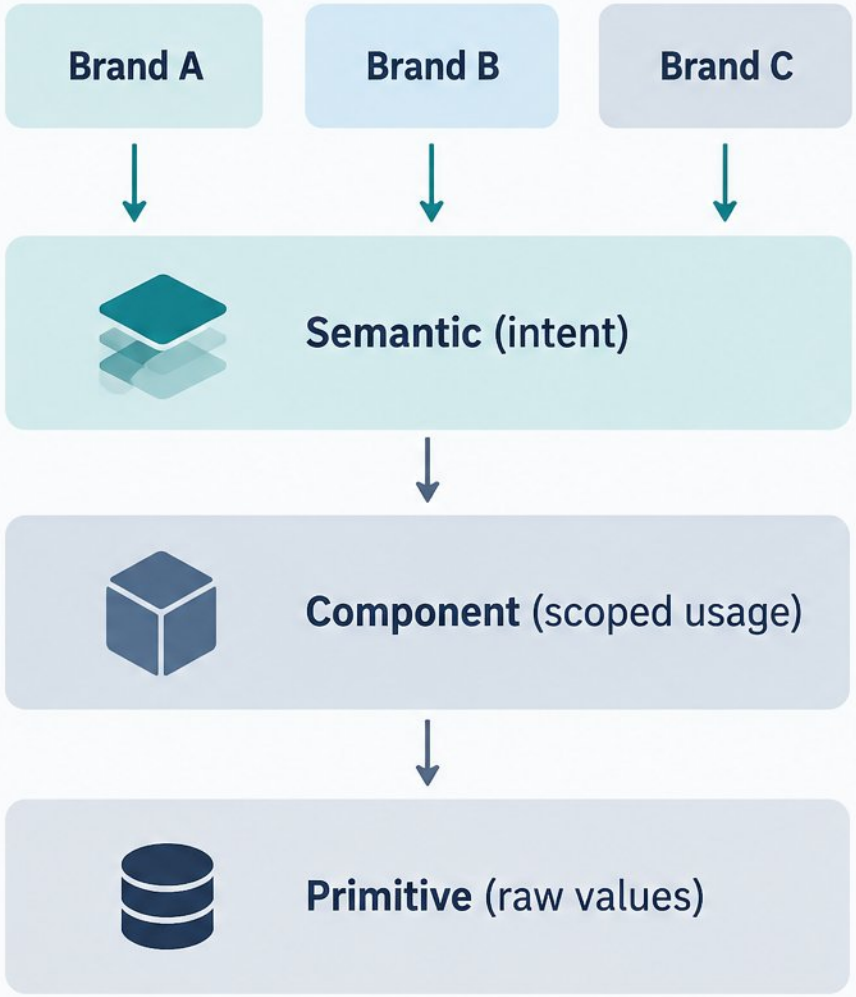
What 'Good' Looks Like Inside a Library

- Required docs: intro, rendered variant and state examples, code API where code exists
- Live rendered components beat screenshots — hover, focus, error, loading stay current
- API discipline: props, states, accessibility behavior, versioning, changelog
- State coverage is the common failure point: default, hover, focus, disabled, loading, empty, error, success
- Build deprecation and migration paths into the release workflow, not ad hoc

Anatomy of a Well-Documented Component



Design Languages and Multi-Brand Token Architecture



- Three tiers: primitive (raw values), semantic (intent), component (scoped usage)
- References flow downward only; component tokens never reach past the semantic layer
- Brands override only the semantic tier, so no component is forked for brand reasons
- Multi-brand: one shared primitive layer; brand files hold aliases only
- Keep the semantic contract small — every new token is a new obligation per brand
- Resolution-order trap: when brand and theme modifiers touch one token, document the order

The Platform Model: Design System as Internal Product

- Ships APIs, tooling, docs, and support tiers—managed with roadmap and SLAs
- Token pipeline backbone: design tool → Git source of truth → transform → versioned packages → docs
- DTCG first stable version (2025.10): theming, multi-brand, P3/Oklch, cross-platform output
- Distribution choices: monorepo vs. multi-package, npm per platform, semantic versioning
- Emerging: Code Connect–style design-to-code linking and CI accessibility/visual-regression gates



Token layers

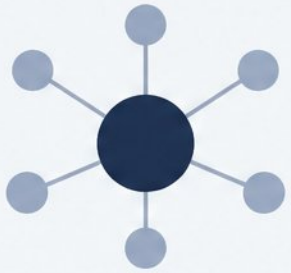
Primitive → Semantic → Component

CI gates

Code Connect–style design-to-code linking and accessibility/visual-regression gates

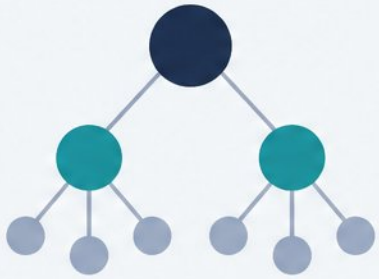
Systems of Systems: Federated and Distributed Models

Centralized



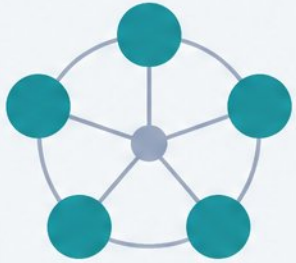
- **Federated:** product representatives co-own and decide together

Hybrid



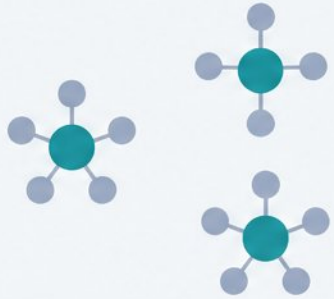
- **Hybrid:** core team owns foundations; domain teams own their patterns

Federated



- **Federated** keeps components grounded in real product needs

Distributed



- **Distributed:** autonomous teams, minimal coordination

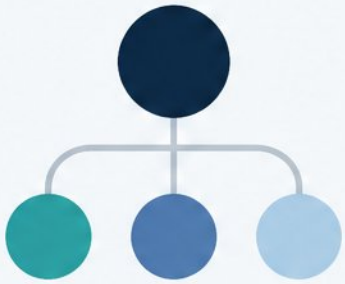
Pipeline

identify → review → spec/sandbox → guild review → build → test → release

Costs: coordination overhead, unclear ownership, unprotected contributor time

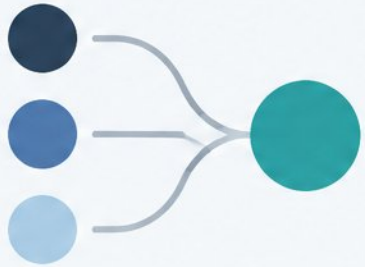
Document the core/domain boundary or hybrid escalates everything

Case Patterns Across System Types



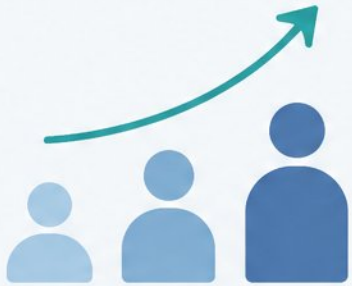
Helios

51 teams, 14 brands, token-based theming; homepage build cut from 3 months to 2 weeks



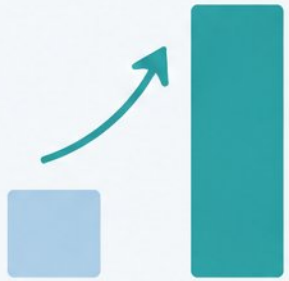
Expedia Group

unified three systems into one token-driven foundation, 50+ components, 4 platforms



Single brand → multi-brand platform

federated RFC model tripled contribution, hit 85% adoption in 18 months



Rollout discipline

adoption 14% → 70% in 12 weeks, fewer detachment and accessibility defects

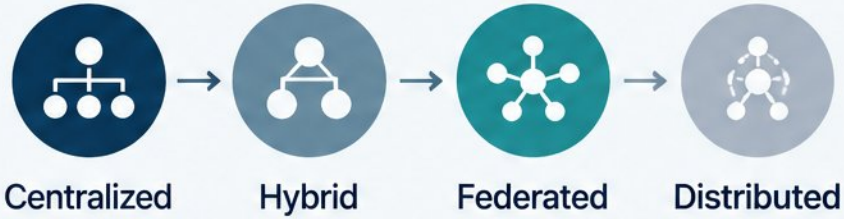


Discussion: which case resembles your context — and what would you change first?

Style guide → System of systems

Governance, Contribution, and Adoption by Type

1 **Match governance to type: centralized early, hybrid to scale, federated with protected capacity**



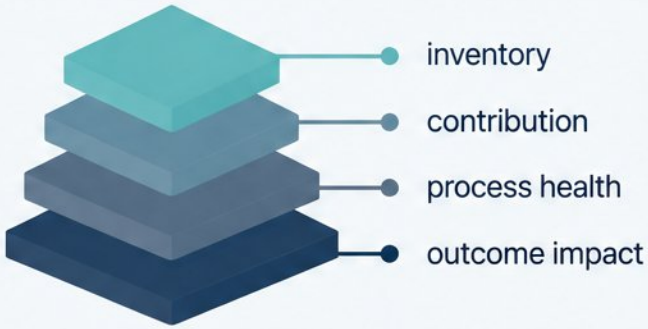
2 **Name the seven decisions:** propose, accept/reject, reuse evidence, a11y check, release, deprecation, exceptions



3 **One test:** is the supported route faster and more predictable than a local fork?



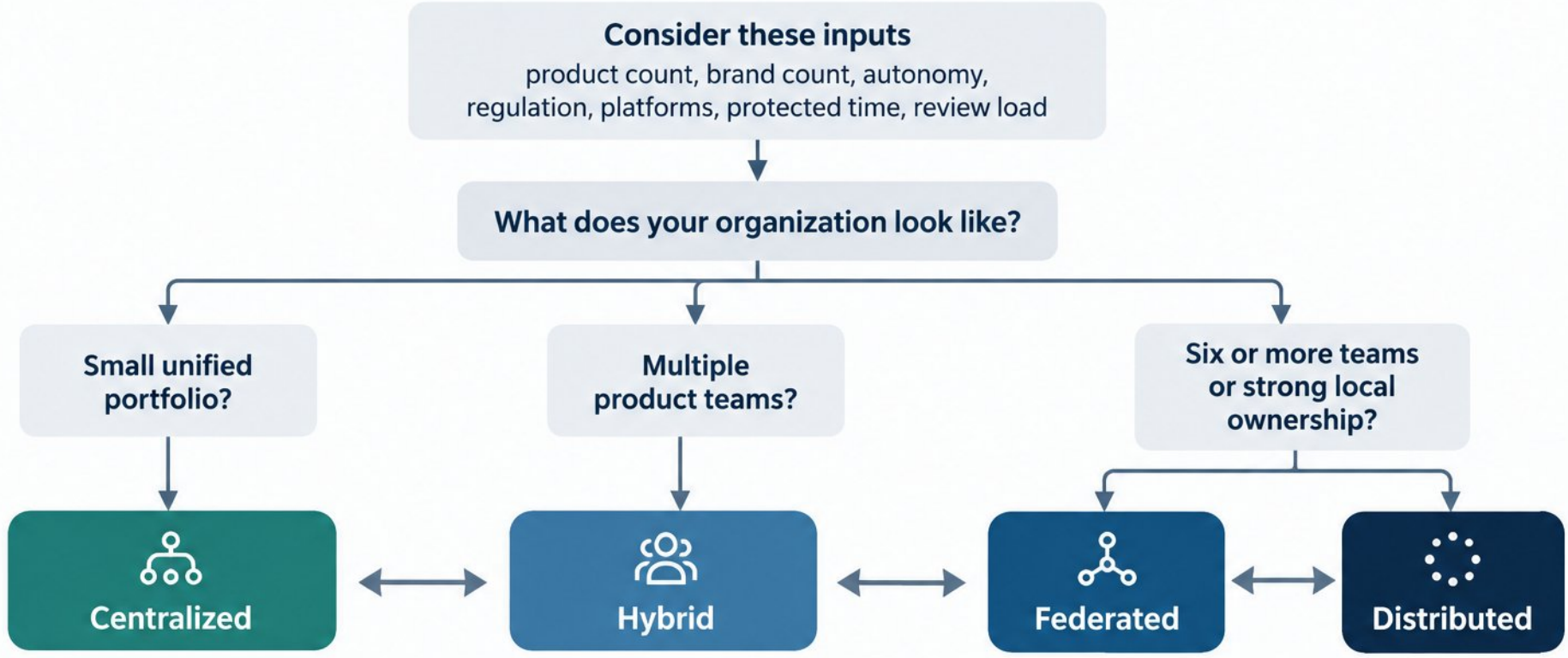
4 **Measure health in four layers:** inventory, contribution, process health, outcome impact



5 **Report counts and rates together; avoid output metrics like components shipped**



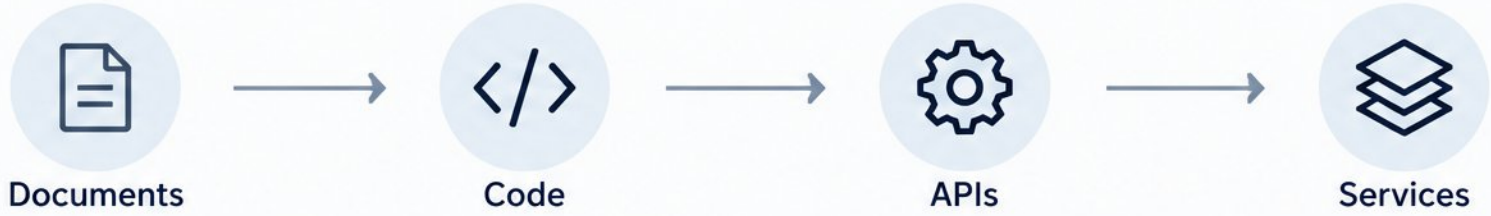
Choosing the Right Type: A Decision Framework



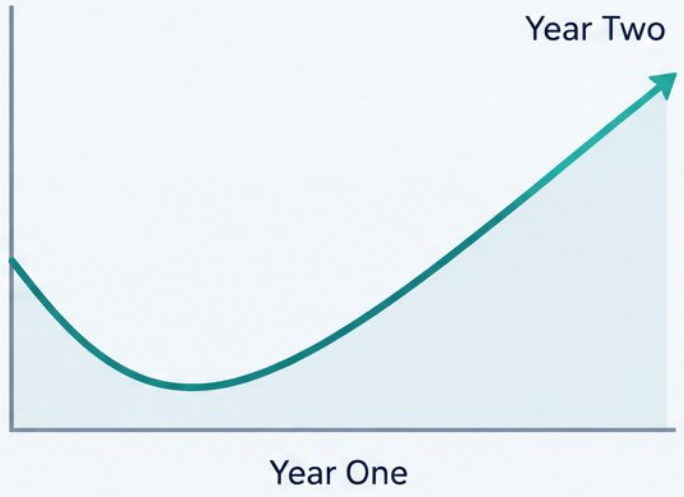
- Inputs: product count, brand count, autonomy, regulation, platforms, protected time, review load
- Small unified portfolio → centralized; multiple product teams → hybrid
- Six or more teams or strong local ownership → federated or distributed
- Risk runs both ways: over-engineering wastes maintenance; under-investing breeds forks
- Migrations work in both directions — plan them, don't drift
- Workshop: score each input, plot current vs. target type, name the gap



Action Plan: Next 90 Days for Your System



- **Audit in 2–3 weeks:** components, token consistency, accessibility, real adoption
- **Turn findings into a cost model:** developer-weeks, duplicated UI, defect rates
- **Pitch finance and engineering together;** frame year one as infrastructure (J-curve)
- **Name an owner,** document one contribution path, pilot it, publish the decision record
- **Take-home:** audit template, DTCG spec, community resource list



Days 0–30 → **Days 30–60: test duplication and cost range** → **Days 61–90: first controlled scope**

PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/2188aee8/public