
PARIVARTA

A CHANGE GOVERNANCE FRAMEWORK

Originated by Asif S. Shaikh | Stewarded by OpenSource Consulting Inc. | v4.0.0

1. Origin & Intent

Parivarta was not designed on a whiteboard in a single sitting, and it is not a rebrand of an existing discipline given a new name. It is the generalization of a methodology refined across ten years and multiple enterprise engagements — proven, independently, at a Fortune 6 global energy leader, a global pharmaceutical leader, and a critical healthcare-logistics provider, among others. What varied client to client was the specific business units and services involved: a pharmaceutical client's Clinical Study Management looks nothing like a healthcare-logistics client's Dispatch and Aviation functions on the surface. What stayed constant, engagement after engagement, was the shape of the lifecycle itself — the same sequence of governance gates, the same decomposition from business service down to IT component. That consistency is the strongest evidence that Parivarta is a reusable discipline, not a bespoke deliverable reinvented from scratch each time.

The critique that produced Parivarta begins with a specific, historical observation about Service Integration and Management (SIAM). SIAM, as commonly practiced, is a derivative of an older discipline — Multivendor Systems Integration (MSI) — with vocabulary borrowed from Service-Oriented Architecture layered on top as branding, rather than as a genuine amalgamation of the two disciplines' actual mechanics. Because "SIAM" became an industry buzzword, few practitioners or clients ever stopped to correct the resulting dysfunction: a framework that manages accountability for suppliers without ever answering what those suppliers' work is actually for. Holding a supplier accountable to a contract is not the same as knowing whether that supplier's output serves a real business outcome — and MSI, under its original name, already accomplished the former. Parivarta exists because the latter question — managing multiple suppliers toward what goal — was never structurally answered by the frameworks practitioners were already using.

There is a second, more personal piece of evidence behind this framework's need to exist independently of any one person. The methodology behind Parivarta was developed and refined over roughly a decade of direct delivery work, including at a large global technology consulting firm, where it was applied across numerous engagements and proof-of-concepts. In the time since that methodology's originator left that firm, it has not sold a further engagement built on it — not because the methodology stopped working, but because the knowledge existed in one person's head and was never fully captured, documented, or productized as an asset the organization could sell independent of that person. That outcome is the clearest possible argument for writing this framework down properly, once, rather than re-explaining it engagement by engagement for another decade.

Parivarta starts from a different foundational question than SIAM or ITIL. Not "how do we hold suppliers accountable," but "how do we transform business objectives into tangible IT outcomes" — the definition of Business Relationship Management (BRM), and the actual foundation Parivarta is built on. Every other element of the framework — the five phases, the Service-Integration-Management structure inside Build, the governance gates — exists in service of answering that BRM question concretely, for a specific client, rather than treating supplier governance as an end in itself.

2. The Five Phases

Every Parivarta engagement moves through the same five phases, in sequence. An engagement can also be scoped to a single phase — a standalone Assess, for instance, delivered as a fit-gap review without proceeding further — when that is what the situation calls for. The five phases are named deliberately so that the discipline is legible on its own terms, the way SIAM, ITIL, and COBIT are each legible as named disciplines independent of any specific vendor's tooling.

Phase	Produces	What Happens
Assess	Current-state maturity picture	A fit-gap assessment of the operating model, business services, service model, and IT model. This is the information used to formulate the plan that follows — Assess does not produce recommendations on its own; it produces the evidence base a Redesign decision requires.

Redesign	Target-state definition	Services are redesigned to conform to defined standards — regulatory, operational, or organizational. Redesign produces the target-state definition that Build then executes against; if Build's work later reveals the Redesign was wrong, that is a signal to revisit Redesign, not to silently patch around it in Build.
Build	The operating model itself	The largest phase by design — see Section 3 for the full structure. Every element of the Service-Integration-Management decomposition, from business service down to individual IT component, is constructed here.
Implement	Live operation	Rollout and Organizational Change Management: training, stakeholder onboarding, and a phased cutover from the current state to the target state defined in Redesign and built in Build.
Sustain	Business-as-usual	Once one or more services are fully defined, the mechanism is complete. Ongoing operation requires minimal further consulting and can run on the client's own BAU resources — this is a deliberate design property, not a limitation. Section 12 covers where AI is intended to extend this phase further.

3. Inside Build: the Service-Integration-Management Structure

Build is deliberately the largest and most structurally dense of the five phases, because it is where the actual operating model gets constructed — everything a client will run on, day to day, once OSCI's direct involvement ends. It decomposes into three nested elements, each of which is itself a substantial body of work.

Service

The Service element covers three stages inherited from, but not limited to, classical service management practice. Service Strategy establishes the service's performance targets, current and future demand, and financial implications — the business case for the service existing in its proposed form. Service Design then designs the service in direct partnership with a Project Management Office, which manages capacity, availability, and network/cybersecurity considerations while acting as the bridge between leadership priorities, budget constraints, and time-to-market pressure. Service Transition follows: the PMO leads a formal transition toward production, encompassing testing, validation, and eventual release.

Integration

Integration is the mapping discipline that connects a designed service to its individual IT components. Where Service defines what a service is meant to do, Integration answers what it actually takes to deliver it — decomposing one business-facing service into the business processes that together deliver it, and each of those processes further into every system, supplier, and dependency required underneath it, so that the service can be visualized as a whole rather than as a collection of disconnected technical tickets. A single Business Service is rarely delivered by one process alone; it is the combined outcome of several — see Figure 2.

Management

Management is the enterprise operating model itself, expressed as a decomposition hierarchy that runs from the organization down to the individual IT service:

- O — Organization (the enterprise as a whole)
- OU — Organizational Unit (a major division — for example, distinct business lines within a single enterprise)
- BU — Business Unit (a function within that division — for example, a specific operational or research group)
- BS — Business Service (a specific service that unit depends on to operate)
- ITS — IT Services supporting that business service, mapped directly from the Integration element above

This decomposition is drawn above, and validated below, as a strict tree — each Business Service belonging to exactly one Business Unit. That is the common case, and the right default. It is not universal: independent enterprise architecture practice (TOGAF, BIZBOK) draws a hard distinction between an organization's reporting hierarchy and its capability/service map precisely because a single business capability routinely spans more than one business unit — TOGAF's own guidance warns explicitly against forcing a tight alignment between business-unit names and top-level capabilities. Parivarta accommodates this directly rather than forcing a false single owner: a Business Service may be declared as co-owned by more than one Business Unit, with every downstream relationship — governance, escalation, People & Roles — correctly reflecting every genuine owner, not just a primary one chosen for convenience.

Nested inside Management sit four further models, each addressing a different governance question. The Service Model is the implementation of a Service Management Office (SMO), holding Incident Management, Problem Management, IT Change Management, SLA Management, Service Catalogue Management, Configuration Management, Availability Management, Service

Continuity Management, Event Management, Release Management, Service Request Management, Capacity Management, and Transformation/Service Improvement Management — the full working discipline that ITIL, on its own, only partially organizes around a business-service lens. The Performance Model measures the IT components identified above against KPIs, SLAs, and KCIs, turning raw operational data into a defensible performance record. The Governance Model assigns roles and responsibilities — Process Owner, Service Owner, Service Coordinator, and others — so that a delivery story reaches leadership in a form that supports data-driven decisions rather than raw, uninterpreted data; this role structure is now formally specified in its own companion document, the Parivarta Accountability Model (PAM), originated in 2016 and independent in its own right — Section 3's decomposition defines what a service is made of, PAM defines who must be accountable for it. The Supplier Model extends beyond typical vendor contract management into day-to-day supplier performance management for each individual service a supplier provides, recognizing that a single supplier may hold one overarching contract with an organization but separate collaboration agreements for each distinct service delivered under it — the same agreement hierarchy PAM defines in full.

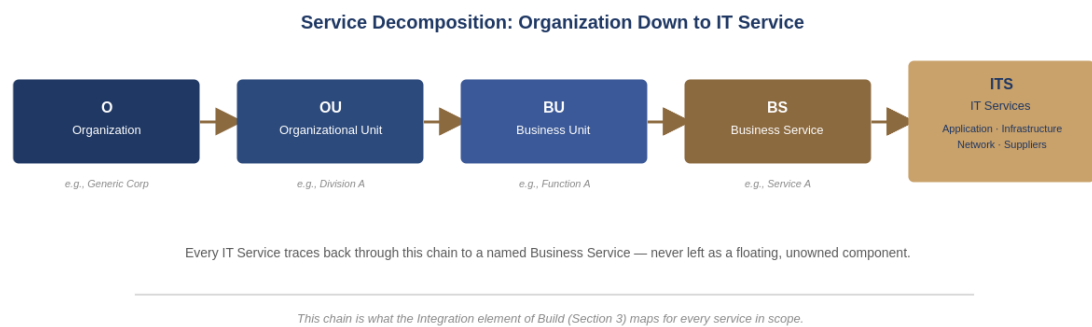


Figure 1 — The O/OU/BU/BS/ITS decomposition chain, applied to any client's actual organization structure.

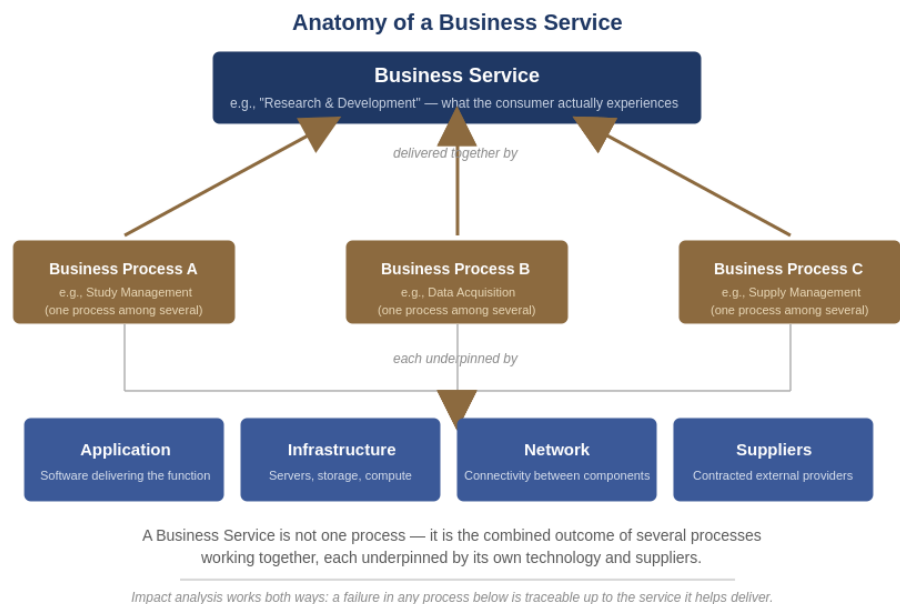


Figure 2 — A Business Service is delivered by multiple Business Processes together, each underpinned by its own technology and suppliers.

The extension named in earlier versions of this document as an unbuilt gap — one component's dependency on another — now has a real design, worth stating precisely rather than left open. Parivarta treats every accountable, changeable thing as a Configuration Item in one unified sense: not only the individual technology component, but the Business Service and Business Process above it, and every Person, Supplier, or Knowledge Article that participates in delivering it. This follows the actual ITIL 4 standard more literally than most CMDB implementations do — a Configuration Item is formally defined as any component that needs to be managed in order to deliver a service, a definition broad enough that ITIL's own typology names Organization CIs (internal policy, strategy) as a category in their own right, and mature practice already extends this routinely to employees and suppliers, not only to hardware and code.

Under this model, every CI — regardless of type — carries a single reference to the CI immediately above it, forming one recursive tree running from the Business Service downward, through Business Process, through IT/Technology Service, to the individual component, person, or knowledge item at the bottom. The consequence is structural, not cosmetic: upstream and

downstream impact analysis becomes a single traversal of this one tree, unconditional on what type of CI is involved. A Person acting as a required resource, a Server, and a Knowledge Article all resolve their business impact through the identical mechanism, rather than through separate logic per type. This is distinct from, and does not replace, the O/OU/BU organizational hierarchy described above — that hierarchy answers who reports to whom; the CI tree answers what exists and what depends on what. The Parivarta Accountability Model, detailed in its own companion document, is what ties the two together — assigning accountable people to specific CIs anywhere in this tree, through one generic mechanism rather than a fixed rule per CI type. As of this version, this unified structure is no longer a design decision — it is the live production schema. A real migration moved every one of OSCI's own 300 components, 55 business processes, and their full escalation history into this exact shape, verified at every step against real, pasted database output rather than assumed. The complete ITIL 4 Service Management practice set now exists as real tables built on this same tree — Incident, Problem, Change, Release, Configuration, Service Level, Asset, Monitoring and Event, Service Request, Service Catalogue, Risk, and Continual Improvement — not a partial subset chosen for convenience.

Authentication now sits on top of this structure as well: real accounts, real password hashing, and a token design deliberately chosen for what a subscription software business specifically needs — the ability to cut off access instantly, not merely let a credential expire on its own schedule over days. A short-lived signed token governs ordinary use; a separate, database-tracked token can be revoked the moment a subscription lapses or an account is compromised, with the cutoff proven, not assumed, by actually revoking a real token in testing and confirming the very next request correctly failed.

Since v2.0.0: The Seven-Module Wheel

The unified CI tree described above is now surfaced through a real, working hub, not just a schema. Every module reaches into the same Parivarta engine at the center — the same CI tree, the same PAM role assignments, the same live Priority Engine — rather than owning a private copy of any of it. The seven modules were renamed from their original working titles once their actual scope made the padded “Service ____ Management” pattern read as bureaucratic rather than confident: Strategy, Operations, Service Management, Suppliers, People, OCM, and Service Desk. Service Management, the flagship, now holds fourteen real process pages — Incident, Problem, Change, Release, Service Request, Risk, Continual Improvement, Configuration Management, Asset Management, Knowledge Management, Availability Management, Service Continuity, Service Level Management, and Service Validation and Testing — each with real create-and-edit forms, not a static mockup. Organizational Change Management remains the one module deliberately left undefined, since nothing yet gives it real shape; that is a decision, not an oversight.

Three further mechanisms now live in the shared center, reused by every module that needs them rather than rebuilt per module. A live-search Component picker, with optional Business Service and Business Process narrowing filters, replaced every raw numeric ID field across the application. A Knowledge Article search surfaces real, Published articles the moment a component is selected while logging an Incident, ranked live by how well the words being typed actually match — closing the loop the original Knowledge Management discussion opened: an article written after one incident should stop the next person from ever needing to call at all. And a generic Task system — real assignment, with Assigned, Expected Completion, and Completed dates, the last two set automatically rather than typed by hand — attaches to any record through a simple related-type reference, proven first on Change Management and Service Continuity's Recovery Procedures, extensible to anything else without new code.

Verified Against the Standard

Every one of the fourteen Service Management processes was checked, field by field, against real, current ServiceNow and ITIL documentation — not assumed from memory. Several places where Truss diverges from the standard turned out to be deliberate improvements rather than gaps: Service Level targets live on the Business Service itself rather than duplicated across every component it touches, avoiding the exact drift problem a per-CI SLA field invites; component dependency is a real, directional relational table rather than a free-text field; and accountability runs through PAM's real role assignments rather than a flat Owner field typed in once and rarely updated. Real gaps the research did surface — Risk Score, Residual Risk, Recovery Time Actual tracking against Recovery Time Objective, Business Criticality as a distinct Business Service attribute — were built, not merely noted.

A Second Real Tenant

Meridian Health Network exists as a second, fully isolated tenant — a complete, imagined healthcare system with a real O/OU/BU/BS/BP structure, roughly eighty components drawn from real data-center vendor categories, genuine Knowledge Articles with real body content, and PAM roles populated across all five role types and three regions. Every person holding a role has a real login account. This exists specifically to prove multi-tenancy holds beyond OSCI's own data — including a deliberately shared Configuration Item spanning three Business Services, and a real Configuration-Item dependency, so the platform's shared-CI and dependency mechanisms have real data to demonstrate against, not just OSCI's history to lean on.

4. The Parivarta Lifecycle

Once built, the operating model does not conclude with a project close-out. It runs as a continuous, closed loop — the defining difference between Parivarta and a conventional consulting engagement, which typically ends when the deliverable is signed off. This lifecycle has been delivered, in this same shape, across multiple independent client engagements, with only the business-unit clusters changing from client to client.

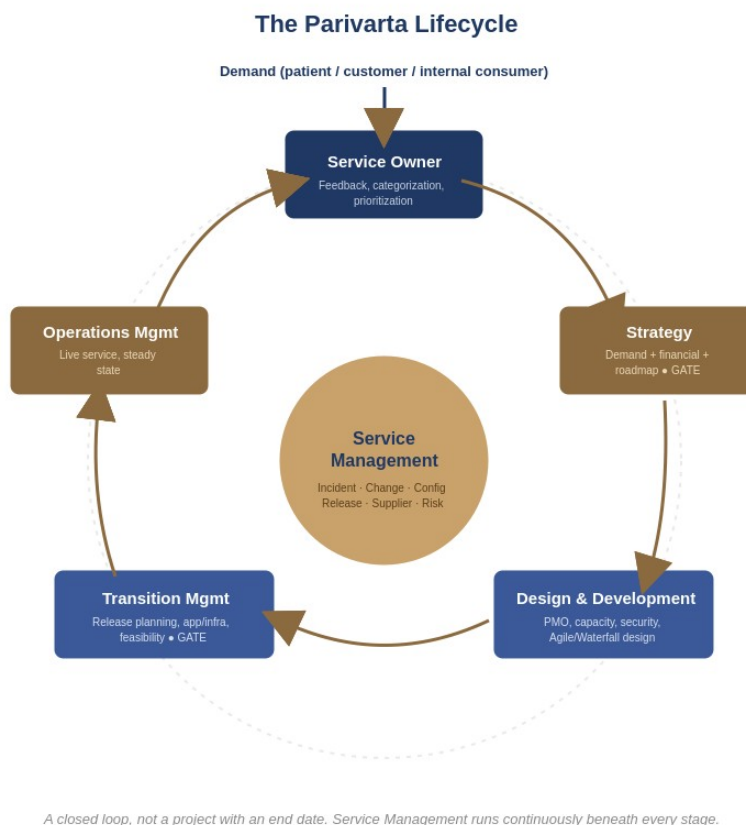
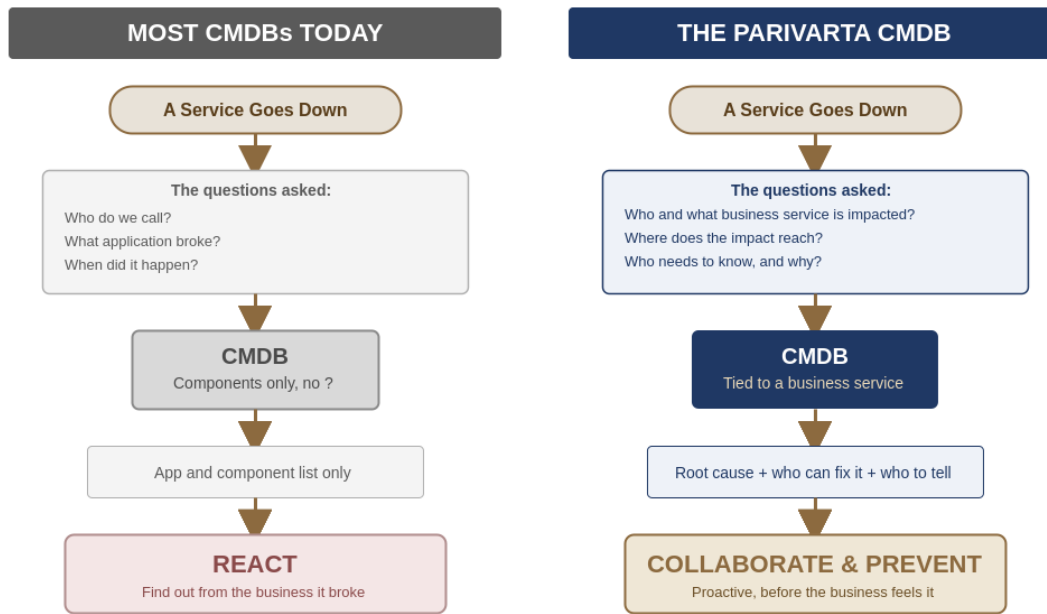


Figure 3 — The five stages run as a closed loop around a continuously operating Service Management core.

- Demand enters through the business — a patient, a customer, or an internal consumer of a service, whose experience is governed by a service-level agreement with the organization.
- Service Owner: customer feedback validation, change categorization, and change prioritization — the first stage where raw feedback is turned into a structured signal.
- Strategy: demand management, financial management, and initiative prioritization/roadmap — the first governance gate, where a named human synthesizes satisfaction, demand, and cost into an actual decision, rather than any one of those three inputs being acted on in isolation.
- Design & Development: PMO oversight, capacity/availability/configuration management, security management, and the design work itself, whether delivered through Agile, Waterfall, or a hybrid method.
- Transition Management: release planning, application management, infrastructure management, and a transition roadmap — validated through testing and feasibility review before anything is released.
- Operations Management: the service goes live and returns to steady-state operation, closing the loop back to Service Reporting and Performance & Corporate Governance.

Underneath the entire loop, Service Management — the ITIL process engine defined in Section 3 — runs continuously, anchored by a Configuration Management Database (CMDB) that ties every component to the business service it supports. This CI-to-service relationship is what gives the model upstream and downstream impact analysis and incident escalation routing, as a direct consequence of the underlying structure rather than as separately engineered features. The same relationship is what makes Configuration Drift a measurable governance risk rather than an abstract concern: because every CI carries a named "Managed By" owner, drift does not require a new detection mechanism to manage — it requires only that ownership be enforced through the same reporting cadence already used for every other governance metric in the model, surfaced at the same recurring review where supplier performance and service metrics are already discussed.



Same underlying event. The difference is what the CMDB is built to answer.

Figure 4 — The same disruption, handled by a component-only CMDB versus one tied to a business service.

5. Performance Model in Practice

The Performance Model described in Section 3 is not an abstract commitment to "measuring things." In practice, it means specific, named metrics with defined formulas, tracked consistently across the ITIL disciplines inside the Service Model. A representative, non-exhaustive set, drawn directly from delivered engagements:

- Incident Management: Mean Time to Resolve (total resolution time across all incidents, divided by incident count), SLA Compliance Rate (incidents resolved within SLA as a percentage of total), and Customer Satisfaction (CSAT) captured via post-resolution survey.
- Change Management: Change Success Rate (successful changes as a percentage of total changes), Change Failure Rate, and Change Lead Time (average time from change request to implementation) — the specific metric referenced in Section 12 as Parivarta's concrete proof point for reduced time-to-market.
- Release Management: Release Success Rate and Release Frequency, tracked to ensure that release velocity is not increasing at the cost of release quality.
- Configuration Management: Accuracy of Configuration Items (the percentage of CIs correctly documented) and Configuration Drift (the percentage of CIs that have deviated from their baseline) — the governance metric referenced in Section 4.

These metrics are not decorative reporting. Each one is designed to be presented at a specific, recurring governance forum — see Section 6 — so that the Performance Model and the Governance Model function as one connected system rather than two separately maintained artifacts. In the Truss prototype, this same Performance Model is what a CIO actually sees day to day: every metric above rolls up into a single, colored 0–100 health score per service, deliberately built to be spoken aloud ("we're at 94%") rather than interpreted from a table — the Performance Model made legible at a glance, not just measured.

6. Governance Cadence

A Performance Model that is measured but never reviewed does not change organizational behavior. Parivarta pairs its metrics with a defined reporting cadence, structured so that the right level of detail reaches the right audience at the right frequency, without every stakeholder drowning in every metric:

Forum	Frequency	Participants & Focus
Service Delivery Review	Weekly / Bi-Weekly	Line management and suppliers, reviewing line and tower-level operational metrics.
Operational Performance & Steering	Bi-Weekly	IT leadership, service owners, service management, and supplier

Review		points of contact, reviewing process and service-level metrics together.
Business Service Performance Review	Monthly	Service owners and business relationship management, reviewing service metrics with business stakeholders directly.
Divisional/Executive Governance Review	Monthly / Quarterly	Senior leadership and direct reports, reviewing contract-level metrics and overall program health.

This structure is deliberately layered rather than flat: an operational issue surfaces weekly, at the level closest to where it occurred, and only escalates to executive governance if it represents a genuine trend rather than a single incident.

7. Governance Gates: Where Humans Decide

Parivarta is built for a world where AI increasingly performs the analysis and drafting work inside this lifecycle — but the accountable decisions remain with named humans, at defined gates, every time. This is not a limitation to be engineered around later; it is a structural feature that predates any AI involvement in the framework at all.

- Strategy Gate — a human validates and audits the synthesized satisfaction, demand, and financial picture before it becomes a decision. Where AI is introduced, its role here is limited to contextualizing feedback and drafting the synthesis report a human reviews — not making the strategy call itself.
- Design Approval — the design document is approved by Strategy, Financial, Demand Management, and Leadership stakeholders before it proceeds. A second, AI-assisted capability under consideration here is a review pass that checks a drafted design or project plan against defined conformity standards and flags gaps before it reaches this human approval step.
- Transition Gate — feasibility and validation are confirmed before implementation begins.
- Release & Change Management — traditional Change Advisory Board (CAB) governance sits before anything reaches production, exactly as it would in an unaugmented, fully manual delivery of this same lifecycle.

This structure matters most in regulated environments — GxP, SOX, and similar frameworks — where an auditor needs to see a defensible chain of accountable human decisions, not a black box. An AI-drafted report that a named human reviewed, corrected if necessary, and approved at a defined gate is auditable in exactly the way traditional CAB governance already is. An AI system that acted autonomously, with no equivalent gate, would not be. Parivarta's AI direction, covered fully in Section 12, is designed from the outset to sit beside these gates, never to replace them.

8. The Priority Engine: Impact × Urgency

This section documents mechanics implemented and stress-tested in Truss, the working prototype of Parivarta, not aspiration. Priority is not assigned by judgment call — it is computed, following the same Impact × Urgency standard used across mature ITIL practice, so that a component failing at 2 a.m. gets the same priority a human would assign it at 2 p.m.

Impact measures breadth: High when a failure crosses more than one Business Service (shared infrastructure — a firewall, a core network link), Medium when it is contained to one service but touches more than one Business Process within it, Low when it is a single, isolated component. Urgency measures time pressure: it is calculated directly from elapsed time since failure, divided by the affected service's SLA target. A ratio at or past 1.0 (the SLA target reached) is Critical; past 0.75 is High; past 0.25 is Medium; below that is Low. The same failure can climb from Low to Critical urgency automatically, without anyone re-touching it, simply by sitting unresolved.

Impact \ Urgency	Critical / High	Medium	Low
High	P1	P2	P2
Medium	P1 / P2	P3	P3
Low	P2 / P3	P4	P4

Each Priority level carries a defined response expectation: P1 pages on-call and opens a major-incident bridge immediately; P2 assigns a senior resource with a 4-hour target; P3 is resolved within the current business day; P4 is logged for resolution within 24–48 hours with no emergency escalation.

9. Two Escalation Lenses, One Reporting Hierarchy

Parivarta uses a single, computed reporting hierarchy — Business Analyst reports to Process Owner, Process Owner reports to Manager, Manager reports to Director — derived directly from who is assigned to which component, process, business unit, and org unit. Nobody re-enters this structure by hand; it falls out of the same assignment data used everywhere else in the framework.

That one hierarchy maps directly onto a second distinction worth naming explicitly: the boundary between Downstream and Upstream. Business Analyst and Process Owner sit at the Business Service / IT Service layer, where the actual operational work happens — this is Downstream, and it is IT-Operational in character. Manager and Director sit at the Business Unit / Organizational Unit layer, which is categorization and accountability metadata, not a layer where work itself occurs — this is Upstream, and it is Business Leadership in character. This is not a technical dependency concept; it is a communication boundary, and it exists to answer one specific question before any message is drafted: does this stay an IT-level conversation, or does it need to speak in business-outcome terms because it has genuinely crossed into leadership's domain.

That one hierarchy is read through two different lenses, depending on the question being asked:

- Incident-level (responding to one failure): escalation is phased by Priority. A P4 stays with the Business Analyst. A P3 reaches the Process Owner. A P2 reaches the Manager. A P1 reaches the Director immediately. This lens crosses Downstream to Upstream exactly once, cleanly, at the P2 threshold — the same point at which a failure stops being purely an IT matter and becomes something Business Leadership needs to know about.
- Process-level (ongoing oversight of a process's health): escalation is staged by SLA breach severity, independent of any single component's individual priority. First contact is the Manager, once the process's SLA target is breached at all. Second, the Process Owner joins once elapsed time reaches twice the SLA target. Third, the Director joins at three times the SLA target. This is the lens used to monitor a process's overall standing, not to respond to one alarm.

The two lenses can disagree in the short term — a single P1 component failure may page a Director immediately under the incident lens, while the process it belongs to has not yet crossed the Manager-contact threshold under the process lens — and that is intentional. One question is "how bad is this specific failure," the other is "how is this process doing overall." Conflating them would hide exactly the information each is designed to surface. Worth noting honestly: the process-level lens, because it begins with Manager rather than Business Analyst, actually crosses Upstream → Downstream → Upstream across its three stages, unlike the incident-level lens's single clean Downstream-to-Upstream crossing. That asymmetry is a direct, known consequence of the process-level lens being designed around business-monitoring cadence rather than incident-response order, and is flagged here as a real design question for the Organizational Change Management and Communication work this Downstream/Upstream boundary is intended to support — not yet resolved as of this version.

10. Problem Management: Recurring-Failure Detection

ITIL distinguishes a Problem (the underlying, recurring cause) from an Incident (one occurrence of its symptom). Parivarta implements this distinction mechanically rather than requiring a separate manual investigation to notice the pattern: any component that fails more than once within the tracked incident history is automatically flagged as a Problem candidate, surfaced against the service it belongs to. A component that fails once is an incident. The same component failing twice is a signal that the incident response, however successful each time, is not addressing the actual cause.

11. Service 360: The Complete Service View

The Priority Engine, the reporting hierarchy, and Problem detection all roll up into a single view per Business Service — built to answer the question a CIO actually asks when looking at one service, not an IT practitioner's checklist. Service 360 shows: component count and Configuration Drift; the defined SLA target; Active Incidents (live, right now) and Resolved Incidents (this engagement's history); Problems (recurring-failure candidates); Changes and Releases against components in that service, where a Change/Release log has been supplied; and every person — named, with title — structurally attached to that service through the reporting hierarchy above.

Two of these — Changes and Releases — depend on a log Parivarta does not generate on its own; where that data has not been supplied, Service 360 says so plainly rather than showing an invented number. Everything else on this view is derived entirely from the CMDB and reporting hierarchy already described in Sections 3 and 9.

12. The Road Ahead: AI-Augmented Sustain

This section describes direction, not current capability, and that distinction is intentional and important. Parivarta has been delivered successfully for a decade using human effort alone — the framework does not depend on AI to function, and every

claim in Sections 1 through 11 above describes work that has already been done, or already built and stress-tested in the Truss prototype, without dependence on AI.

The structure this direction builds on is not new — it is the same operating wheel Parivarta's originator designed in 2018, now seven modules following a 2026 renaming, shown below with an honest status against what exists in Truss and this Framework today.



Figure 5 — The Parivarta Wheel: Service Management fully built across fourteen real process pages, three modules partially built, one deliberately deferred rather than silently missing.

The stated intent is for AI to act as a performance enhancer inside the Sustain phase specifically, in two concrete, bounded ways rather than as an open-ended autonomous system. First, an analysis and summarization capability: ingesting CSAT and service-delivery data, and producing the contextualized report a human reviews at the Strategy Gate — a task that is well within the reliable range of current AI capability, since it involves pattern-matching and summarizing structured data rather than generating unbounded new content. Second, a review capability: checking a drafted strategy document or project plan against defined conformity criteria and flagging errors or gaps before it reaches a human decision-maker at the Design Approval step. Both capabilities sit next to a specific human gate; neither replaces one.

The goal of this direction is a measurably shorter Change Lead Time — the same metric already defined in Section 5, already measured in delivered engagements today. The pitch to a future client is accordingly not speculative: it is "here is your Change Lead Time today, measured with the same framework used in Build; here is what it looks like once the Sustain-phase AI layer is added," using a metric the client will already trust because it is the same one used throughout delivery.

Longer-term, this direction points toward a lightweight application — deliberately not a rebuild of existing ITSM platforms such as ServiceNow or Remedy. The core insight behind that scoping decision is that the essential function of any such platform is its Configuration Management Database: a record of what exists, tied to who owns it, what it connects to, and when it was last validated. Everything else is substantially manipulation of fields around that core. Parivarta's product direction is therefore to own that CMDB-and-service-mapping core, purpose-built around the O/OU/BU/BS/ITS structure defined in Section 3, while assuming a client's existing ITSM tool continues to own transactional ticketing — pulled in via integration rather than rebuilt from scratch. The governance gates in Section 7 form the product's core interaction model: not a configurable workflow engine, but an opinionated implementation of a proven methodology.

The initial path to that product is consulting-led, not product-led: an engagement's Build phase is also, functionally, the data-preparation work — clean CMDB, populated service mapping, defined KPIs — that a future AI-augmented Sustain layer would

require to be useful at all, in the same way that ServiceNow's own AI capabilities are only as effective as the CMDB and knowledge-base data underneath them. A client who has been through Build with OSCI is, by construction, the client best positioned to adopt the product first.

13. Onboarding Is Governance: The Setup Lifecycle

Every governance framework assumes the organization it governs already has a correct structure. In practice, that structure is usually the least governed thing in the building: it is loaded once from spreadsheets, nobody signs it off, and it starts drifting the day it goes live. Parivarta treats the initial setup of an organization as a governed change in its own right, and the largest single piece of an OSCI engagement.

The people who design the structure are the people who will approve every change to it afterwards. Business leaders lay out their own Organization, Org Units, Business Units, Business Services, Business Processes, and Process Steps. IT leaders map IT Services onto those steps. People and their accountability roles are added last. The OSCI consultant guides the work, tests the result with leadership, and records their sign-off. The whole organization moves through four states:

- **In Setup.** Every phase is being built and is freely editable. Work is staged in batches that go from Draft to In Review to Approved before they are committed.
- **In Review.** The consultant walks leadership through the complete setup and tests it with them.
- **Approved.** Leadership sign-off is recorded: who approved, their title, the date, and confirmation that testing was done. The consultant records it on their behalf and the record shows who entered it.
- **Live.** Setup closes. From this point, the only way to change the structure is through Service Management, by Change or Service Request, approved by the same leaders who signed it off. Consultant access ends automatically.

This is the same principle as the governance gates in Section 7, applied one step earlier. A human decides when the organization is correct, the decision is recorded, and nothing changes afterwards without passing through a gate.

14. Business Services and IT Services: Parallel, Not Nested

Earlier versions of this framework drew the decomposition as one chain, from Organization down to IT component. Real engagements showed that the business side and the technology side are two separate structures that meet at a defined seam, not one structure where technology hangs underneath business.

- **A Business Service** A collection of business processes that delivers an agreed outcome to the business. Its commitment is a Service Level Agreement (SLA).
- **An IT Service** A collection of IT processes, underpinned by technology, that delivers an agreed outcome. Its commitment is an Operational Level Agreement (OLA). It may be delivered directly through IT Processes, or through Products where real vendor variety sits underneath (for example, NAS delivered by a specific vendor's NAS).
- **The seam.** Each Business Process is made of named, ordered Process Steps. Enterprise processes commonly have one or two hundred. Each step depends on exactly one IT Service. This is where business and technology meet, and it is how an outage on a technology component can be traced to the exact business step it stops.
- **Suppliers.** A supplier is bound by an Underpinning Contract, with one Collaboration Agreement for each IT Service or Product it delivers. Coordination roles can attach to the Collaboration Agreement itself.

The IT Service Catalog is described in five questions, in the order clients have consistently recognised: WHO is accountable (from PAM roles, at two levels: the business view and the support view), WHAT the service is, WHEN it is available and what to do when it is not, WHERE it is offered (by region), and HOW it is delivered (its IT Processes and Products). The catalog is built once and displayed two ways: a business catalog that shows outcomes and who to call, and a technical catalog that shows the delivery chain underneath.

15. Accountability Decides Access and Routing

If accountability is real and recorded, it should decide two things that most tools maintain separately and by hand: what a person can see and do, and where work is sent.

- **Access follows accountability.** A person's access comes only from their PAM roles, scoped to each role's target. No role means no access. Holding several roles gives the union of their scopes. Financial visibility is a separate, explicit flag on a role, never implied. Named profiles bundle common role combinations so they can be applied consistently.

- **Routing follows accountability.** Instead of assignment groups that go stale as people move, work is routed to the Analysts on the affected component, walking up the tree to the nearest level that has them, and escalated along the PAM ladder from Analyst to Management, Ownership, and Directorship. This is designed and recorded as a challenge Truss addresses; it is not yet built.

16. Why Parivarta, Not SIAM or ITIL Alone

- SIAM answers who is accountable for a supplier. Parivarta answers what that supplier's work is for, by tying every component back through Integration to a named business service, all the way up to the Organization itself.
- ITIL governs process discipline inside IT. Parivarta starts one level higher, at Business Relationship Management, and uses ITIL's own disciplines — inside the Service Model — as one working part of a larger structure, rather than treating them as the whole of the discipline.
- Neither SIAM nor ITIL defines where a human must decide versus where a system may act. Parivarta's governance gates make that distinction explicit from the outset — which is what makes AI-augmentation possible later without losing auditability now.
- Established enterprise platforms already offer significant capability across these same functions — but split across separately priced, separately configured modules that a customer must integrate themselves. Parivarta's differentiation is coherence, not raw capability: one opinionated methodology, expressed as one connected structure, in place of several powerful but disconnected tools a client is left to assemble on their own.