Sensaka

A PROFESSIONAL GUIDE FROM SENSAKA

From Infrastructure Monitoring to Business Service Intelligence

How Sensaka DCOS, iDCOS, and SmartBSM create end-to-end operational visibility.

Enterprise monitoring has become highly effective at collecting technical evidence. Infrastructure teams can observe processor load, memory pressure, disk status, network traffic, storage latency, power consumption, temperature, process availability, database response time, application errors, and thousands of other indicators. Yet during a major incident, senior management still asks questions that conventional monitoring tools often cannot answer: Which business service is affected? How severe is the customer impact? Where did the failure begin? Which team should act first? Has service actually been restored?

This gap exists because infrastructure monitoring and business service intelligence solve different problems. Monitoring determines whether individual resources are available and behaving within expected thresholds. Business service intelligence explains how those resources work together to deliver a business outcome, how technical conditions propagate through dependencies, and which operational decision should follow.

The distinction becomes critical in hybrid and heterogeneous environments. A single customer-facing or internal service may depend on physical servers, storage arrays, network paths, power and cooling, virtualization, cloud resources, operating systems, databases, middleware, containers, APIs, and third-party services. Each domain can appear healthy in isolation while the end-to-end service degrades. Conversely, a high-severity technical alarm may have no immediate business consequence because redundancy is functioning as designed.

Moving from monitoring to business service intelligence therefore requires more than another dashboard. It requires a connected operating model built on trustworthy resource identity, dynamic relationships, normalized events, configuration and change context, service ownership, health logic, impact analysis, governed workflows, and continuous operational learning.

This guide defines that operating model and explains how Sensaka DCOS, iDCOS, and SmartBSM combine physical infrastructure evidence, operational governance, and service-aware intelligence in one end-to-end architecture.

FIGURE 1 — FROM TECHNICAL SIGNALS TO BUSINESS DECISIONS

Business service intelligence converts infrastructure evidence into prioritized, explainable, and actionable business context.

1. Why Infrastructure Monitoring Alone Reaches an Operational Ceiling

Traditional infrastructure monitoring is essential. Without accurate telemetry and event collection, no higher-level analysis can be trusted. The limitation appears when technical observations remain separated by tool, domain, and ownership boundary.

Device severity is not business priority

A critical hardware alarm on a development server may have limited urgency, while moderate storage latency affecting a payment, manufacturing, healthcare, or trading service may require immediate escalation. Technical severity describes the condition of a resource; operational priority should describe the consequence for the organization.

When the two are treated as equivalent, teams either overreact to low-impact events or under-prioritize conditions that are already degrading a critical service.

Domain tools explain components, not service outcomes

Server, storage, network, facilities, cloud, database, and application tools are optimized for their own technical domains. They can provide detailed evidence, but they rarely share a complete model of the service being delivered.

During an incident, engineers must manually compare timestamps, screenshots, topology diagrams, change records, and user reports. The organization effectively asks people to become the integration layer between monitoring systems.

Static documentation cannot represent operational reality

Service maps created in documents or manually maintained CMDB records become inaccurate as virtual machines move, applications are released, components are replaced, cloud resources scale, network routes change, and ownership shifts. A relationship model that was correct during design may not reflect the environment at the time of failure.

Business service intelligence requires continuously reconciled relationships rather than diagrams that are updated only before audits or major projects.

Shared infrastructure complicates fault interpretation

A storage array, network link, database cluster, virtualization platform, power circuit, or cooling zone may support multiple services. One technical condition can therefore have different consequences across business processes. Redundancy, workload distribution, maintenance state, transaction volume, and service criticality all affect the actual impact.

Monitoring success is often measured too narrowly

Collection coverage, polling success, dashboard availability, and alert volume indicate whether monitoring infrastructure is functioning. They do not prove that the organization can identify the affected service, isolate the initiating condition, assign the correct team, restore service efficiently, or prevent recurrence.

FIGURE 2 — THE MONITORING-TO-INTELLIGENCE GAP

	Isolated monitoring	Integrated operations	Business service intelligence
Telemetry & events	✓	✓	✓
Resource identity	Partial	✓	✓
Topology	✕	Partial	✓
Change context	✕	✓	✓
Service health	✕	Partial	✓
Impact & root cause	✕	✕	✓
Workflow & learning	✕	Partial	✓

Monitoring provides evidence; business service intelligence adds relationships, consequence, and decision context.

2. What Business Service Intelligence Actually Means

Business service intelligence is the capability to represent how technology resources collectively deliver a service, assess service health from multiple forms of evidence, explain the likely origin and propagation of abnormal conditions, and direct an appropriate operational response.

It is not simply a business-facing visualization of infrastructure alerts. A dashboard that places red, amber, and green indicators beside application names remains a presentation layer unless the underlying service model is accurate, explainable, and operationally connected.

Five elements of a decision-grade service model

Resource identity establishes which physical, virtual, cloud, software, and business objects exist. It prevents duplicate or ambiguous records from distorting health and topology.

Relationships describe physical, logical, runtime, ownership, and service dependencies. They explain how an abnormal condition can propagate and which resources share a common dependency.

Operational state combines availability, performance, events, configuration, capacity, environmental condition, maintenance, and recent change rather than relying on one threshold.

Service objectives define criticality, user population, operating period, service level, recovery requirement, redundancy expectation, and business owner. These factors determine consequence and priority.

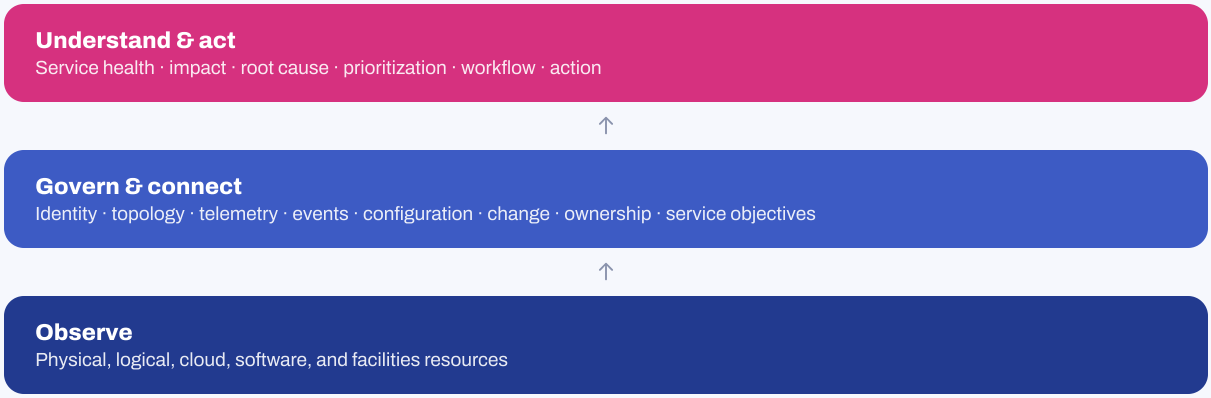
Decision logic converts evidence and context into health, impact, root-cause candidates, ownership, escalation, recommended action, and verification criteria.

A mature platform should be able to answer six operational questions consistently:

1. What service is affected?
2. Which users, locations, transactions, or business processes are at risk?
3. Which resource or dependency is the most credible initiating cause?
4. What recent change or environmental condition may explain the behavior?
5. Which team and controlled procedure should respond?

6. What evidence confirms that service has recovered?

FIGURE 3 — THE BUSINESS SERVICE INTELLIGENCE STACK



Decision-grade service intelligence depends on both technical evidence and governed operational context.

3. Establishing the Data Foundation

Business service intelligence cannot compensate for poor resource data. If the same server appears as several records, if a storage volume is not connected to its consuming host, or if application ownership is unknown, downstream analysis will be unreliable regardless of analytical sophistication.

Canonical resource identity

Every managed object should have a stable identity and provenance. Hardware identifiers, serial numbers, management-controller IDs, cloud resource IDs, platform-specific unique IDs, chassis relationships, and software instance identifiers can be reconciled into canonical records. Hostnames and IP addresses should generally be treated as mutable attributes rather than sole identity keys.

The platform should retain source evidence, matching confidence, conflicts, and historical values. Ambiguous records require governed review instead of uncontrolled automatic merging.

Cross-domain topology

The relationship model should connect facilities, racks, devices, components, interfaces, network paths, storage objects, hypervisors, virtual machines, cloud resources, operating systems, databases, middleware, containers, applications, and business services.

Not all relationships represent the same dependency. The model should distinguish physical containment, network connection, data path, runtime hosting, cluster membership, service consumption, ownership, redundancy, and conditional failover. This semantic precision improves both impact analysis and root-cause reasoning.

Normalized operational evidence

Events from different tools and vendors need a common structure: source, component, condition, severity, timestamp, status, recovery, evidence, and confidence. Performance and capacity metrics require consistent units and time alignment. Configuration and environmental data should also be connected to the same resource identity.

Normalization must preserve original evidence. Engineers and vendors may still need the native event code, sensor value, log entry, or device response during diagnosis and escalation.

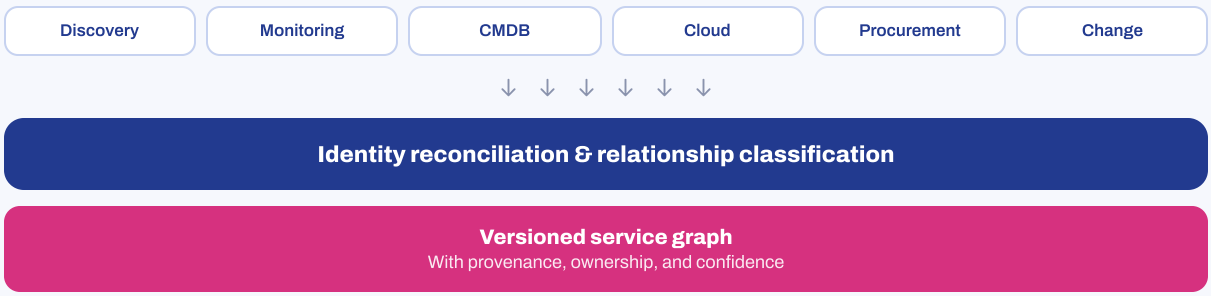
Change and maintenance context

Recent releases, firmware upgrades, configuration changes, component replacements, workload migrations, network changes, and approved maintenance windows often explain new behavior. Service intelligence should therefore incorporate change records and detected configuration drift directly into incident analysis.

Ownership and service objectives

Each critical service needs an accountable owner, support team, escalation path, operating calendar, service level, recovery objective, user or location scope, and critical business periods. Without this context, the platform can describe technical health but cannot determine business priority.

FIGURE 4 — FROM DISCONNECTED RECORDS TO A GOVERNED SERVICE GRAPH



A governed service graph is the analytical foundation for health, impact, and root-cause decisions.

4. Converting Events into Service Health

Service health should not be calculated by simply taking the most severe alarm from any connected resource. That method makes large services permanently appear unhealthy and gives downstream symptoms the same weight as initiating faults.

Normalize and deduplicate evidence

The same condition may be reported by a hardware controller, operating system, vendor console, SNMP trap, syslog, database monitor, application monitor, and synthetic test. Equivalent observations should be grouped into one evidence set while retaining their original sources.

Deduplication reduces noise; it does not discard diagnostic detail.

Correlate through topology and time

Events occurring on related resources within a meaningful sequence can be grouped into an incident candidate. A failed power supply may reduce hardware redundancy without immediate application impact. A storage path failure may trigger host, database, middleware, and application symptoms. Network packet loss may appear as database latency even when the database itself is healthy.

Topology explains where propagation is possible; time helps determine which condition appeared first.

Model degradation, not only availability

A service can remain technically online while throughput declines, latency increases, transaction errors rise, a cluster loses redundancy, or capacity approaches an operational limit. Health models should therefore represent healthy, at risk, degraded, partially unavailable, and unavailable states as appropriate for the service.

The health policy should combine technical signals with service-specific thresholds, redundancy, workload, business period, and recovery objectives.

Calculate business impact

Impact analysis should identify affected services, users, sites, transactions, dependent processes, and service commitments. It should also distinguish confirmed impact from potential exposure. A failed redundant component may create risk without current interruption; a shared dependency failure may affect several services simultaneously.

Rank root-cause candidates with evidence

Root-cause analysis should produce explainable candidates rather than an unexplained conclusion. Relevant evidence includes event sequence, topology position, shared dependency, configuration change, maintenance state, metric deviation, historical pattern, and recovery behavior.

Confidence should be visible. Where evidence is incomplete, the platform should guide additional checks rather than present certainty that cannot be justified.

FIGURE 5 — EVENT-TO-SERVICE INTELLIGENCE PIPELINE

Service intelligence emerges when technical evidence is correlated with relationships, change, and business objectives.

5. Designing Business Service Models That Remain Useful

Many service-mapping initiatives fail because they attempt to model the entire enterprise before delivering operational value. The resulting project becomes too broad to govern, and the topology is outdated before it is used in a real incident.

Start with critical operational questions

Select services where faster diagnosis, stronger continuity, regulatory evidence, or business-impact visibility has clear value. Define what the organization needs to decide during degradation: which business process is affected, what dependencies are critical, what redundancy should exist, who owns each layer, and which recovery evidence is required.

The service model should be designed to answer these questions, not to document every theoretical relationship.

Combine top-down and bottom-up modeling

Top-down modeling begins with the business service, owner, service objective, application, transaction, and user journey. Bottom-up discovery identifies the infrastructure, software, and runtime dependencies actually observed in the environment.

The two views should be reconciled. Top-down design provides business meaning; bottom-up evidence prevents the model from becoming an architectural assumption.

Represent shared and conditional dependencies

Critical services often share storage, networks, virtualization clusters, databases, identity platforms, power, or cooling. They may also use primary and secondary paths, active-active clusters, failover resources, or dependencies that are relevant only during certain processes.

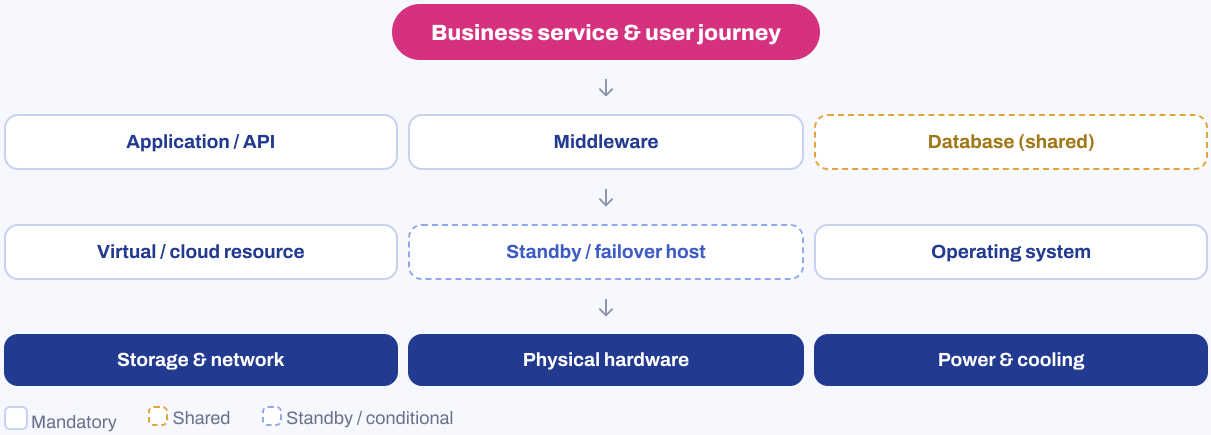
The model should distinguish mandatory, redundant, shared, standby, and conditional relationships. Otherwise, impact will either be overstated or missed.

Govern model lifecycle

Service models require ownership, versioning, freshness indicators, reconciliation rules, and periodic validation. Detected changes should trigger review when they affect a critical relationship. Retired resources and services should be closed consistently so they do not

remain in active health calculations.

FIGURE 6 — A SERVICE DEPENDENCY MODEL FOR A CRITICAL APPLICATION



Accurate impact analysis requires semantic relationships, not merely a visual topology.

6. Embedding Intelligence into the Incident Lifecycle

Business service intelligence delivers value only when it improves operational execution. Health, topology, and root-cause analysis must therefore connect directly to incident, change, knowledge, approval, automation, and verification processes.

Detection and triage

When an abnormal condition is detected, the platform should enrich it with affected resource, service, topology, owner, location, configuration, recent change, warranty, maintenance state, and relevant knowledge. Related symptoms should be grouped so responders begin with one coherent incident rather than hundreds of independent alerts.

Assignment and escalation

Routing should use likely fault domain, service ownership, technical responsibility, location, criticality, and support coverage. Major-incident escalation should be triggered by confirmed or probable business impact, not solely by the number of alerts.

Diagnosis and remediation

Responders should receive an evidence trail showing event sequence, relationship path, metric deviation, change context, and root-cause candidates. Approved diagnostic and remediation procedures can then be recommended or executed through governed workflows.

Automation should validate prerequisites, enforce authorization and approval, capture action parameters, preserve audit evidence, verify results, and stop safely when the environment differs from expected conditions.

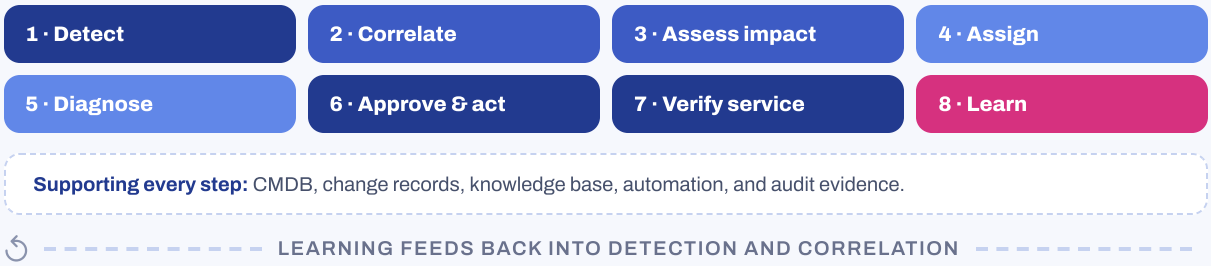
Service recovery verification

Closing a device alarm does not prove that service has recovered. Verification should confirm resource status, dependency health, transaction behavior, user-facing indicators, redundancy, and the service objective. The incident can then move from technical recovery to validated business restoration.

Operational learning

Post-incident review should improve event rules, service relationships, health policies, knowledge, automation, capacity thresholds, supplier evidence, and architectural risk controls. The platform becomes more effective when resolved incidents strengthen the operating model rather than disappear into closed tickets.

FIGURE 7 — SERVICE-AWARE INCIDENT RESPONSE LOOP



Service intelligence should guide the complete incident lifecycle, from evidence collection to validated recovery and learning.

7. High-Value Operational Scenarios

The following scenarios demonstrate how service context changes the quality of operational decisions.

An MES slows down while every system remains online

Infrastructure dashboards may show that servers, databases, storage, and networks are available, yet production throughput falls and work-in-progress accumulates. A service-aware model connects the MES transaction path to middleware queues, database calls, virtual hosts, storage latency, and shop-floor network segments.

Instead of debating which team owns the problem, responders can identify the dependency whose behavior changed first, determine which production lines and transactions are affected, and validate recovery against MES processing time rather than device availability alone.

Storage latency appears as an application problem

A shared array or SAN path can degrade several applications at different rates. Database teams may observe slow queries, application teams may see timeout errors, and infrastructure teams may receive path or controller warnings.

Cross-domain topology and time correlation group these symptoms, identify the common storage dependency, and prioritize response according to the services and service levels affected.

A hardware component degrades before service interruption

A power supply, fan, memory module, disk, controller, or interface may enter a degraded state while the operating system and application remain available. Component-level evidence identifies the condition early. The service model then shows whether redundancy remains, which service depends on the device, whether the current business period permits maintenance, and which controlled action should follow.

Facilities risk threatens a group of services

Rising rack temperature, abnormal power draw, PDU loading, or cooling degradation may affect multiple devices without producing immediate application alarms. By associating facilities data with racks, equipment, and services, the organization can identify the exposure before

widespread interruption and coordinate workload, capacity, and maintenance decisions.

A recent change creates a multi-layer symptom chain

A network configuration change, firmware update, component replacement, application release, or workload migration may trigger symptoms across several domains. Connecting detected change and approved change records to topology and incident evidence helps distinguish coincidence from plausible causation and accelerates rollback or corrective action.

SCENARIO	CONVENTIONAL MONITORING VIEW	BUSINESS SERVICE INTELLIGENCE VIEW
MES performance degradation	Separate infrastructure and application metrics	Transaction path, affected production process, initiating dependency
Storage latency	Alerts across array, host, database, and application	Shared storage dependency and impacted services
Component degradation	Hardware warning without business context	Redundancy, service criticality, maintenance priority
Thermal or power risk	Facilities alarm isolated from IT	Exposed racks, devices, services, and capacity options
Change-related incident	Event sequence compared manually	Change, topology, symptom propagation, and rollback evidence

8. Measuring Business Service Intelligence

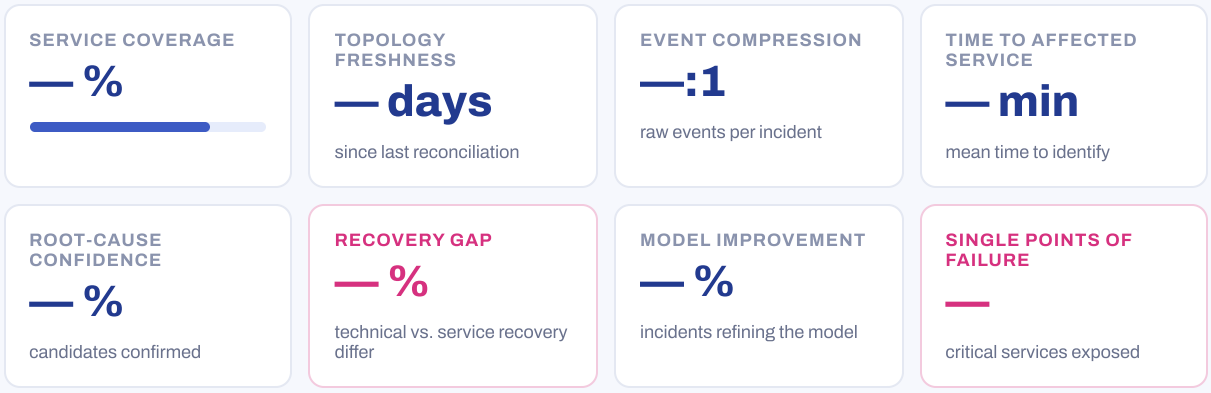
Metrics should determine whether the operating model improves decision quality and service outcomes. Alert volume alone is not a meaningful success measure; a reduction can indicate better correlation or simply reduced visibility.

Recommended indicators include:

- Percentage of critical services with an approved owner, objective, and dependency model
- Percentage of service relationships discovered or reconciled within the required freshness period
- Percentage of infrastructure and software resources matched to a canonical identity
- Percentage of incidents automatically enriched with service, topology, ownership, change, and maintenance context
- Ratio of raw events to correlated incident candidates
- Mean time to identify the affected service and responsible technical domain
- Mean time to isolate a credible root-cause candidate
- Mean time to restore and validate the business service
- Percentage of incidents where technical recovery and service recovery differ materially
- Accuracy of impact predictions compared with confirmed incident outcomes
- Percentage of automated or remote actions executed through governed workflows
- Percentage of major incidents that improve service models, knowledge, or automation
- Number of critical services dependent on unsupported, over-capacity, or single-point-of-failure resources

Metrics should be segmented by service criticality, technical domain, site, business period, and incident category. The purpose is to expose control gaps and guide investment, not to create a single score that conceals operational complexity.

FIGURE 8 — BUSINESS SERVICE INTELLIGENCE SCORECARD (ILLUSTRATIVE)



Effective measurement links data quality and analytical performance to service restoration and risk reduction.

9. A Phased Implementation Roadmap

Business service intelligence should be implemented as a controlled operating-model program. Attempting to map every service and integrate every data source before demonstrating value creates unnecessary scope and governance risk.

Phase 1 — Establish trustworthy technical evidence. Select one critical service and verify monitoring coverage across hardware, storage, network, facilities, software, cloud, and application layers. Reconcile resource identity and identify data gaps.

Phase 2 — Build the service dependency model. Define the business owner, users, objectives, critical periods, applications, technical dependencies, redundancy, and recovery evidence. Combine architectural input with discovered relationships.

Phase 3 — Normalize events and integrate operational context. Standardize event semantics, deduplicate repeated evidence, and connect CMDB, ownership, change, maintenance, knowledge, and incident workflows.

Phase 4 — Introduce health, impact, and root-cause analysis. Configure service-specific health policies, correlation logic, impact calculation, and explainable root-cause candidates. Validate results against real incidents and controlled tests.

Phase 5 — Govern action and continuous improvement. Add approved diagnostic procedures, remote operations, automation, service-recovery validation, post-incident learning, and management reporting.

MATURITY STAGE	PRIMARY CAPABILITY	OPERATIONAL OUTCOME
Observe	Cross-domain telemetry and events	Reliable technical evidence
Connect	Canonical identity and service relationships	End-to-end dependency visibility
Understand	Health, impact, change, and root-cause analysis	Faster, explainable decisions
Act	ITSM, approval, knowledge, and automation	Controlled and consistent response
Learn	Outcome feedback and model refinement	Continuously improving service operations

Implementation should expand service by service. Each new model should inherit standard resource classes, event semantics, governance, and workflow patterns while remaining sensitive to the service's actual architecture and business objectives.

10. How Sensaka Delivers End-to-End Operational Visibility

Sensaka combines three complementary products. DCOS establishes detailed physical infrastructure evidence, iDCOS provides governed operational context and execution, and SmartBSM transforms relationships and evidence into business service intelligence.

Sensaka DCOS: physical infrastructure evidence below the operating system

DCOS discovers and monitors heterogeneous servers, storage, network and security devices, racks, power systems, environmental equipment, and hardware components. It supports out-of-band and agentless collection, detailed configuration, component health, detected change, asset and location data, warranty, energy, environmental condition, capacity, and governed remote operations.

This independent hardware perspective remains available when the production operating system or application is unavailable. It also reveals component degradation, power, temperature, and configuration conditions that application-centric monitoring may not observe.

For business service intelligence, DCOS contributes the physical evidence needed to connect service symptoms to the actual infrastructure supporting them.

Sensaka iDCOS: operational governance and execution context

iDCOS manages software and infrastructure resources through CMDB, ITSM, knowledge, approval, orchestration, scripts, and automated tasks. It supports configuration items, relationship management, topology, incidents, problems, changes, releases, service levels, ownership, and controlled workflows.

iDCOS provides the governed context required to determine who owns a service or resource, what changed, which maintenance is authorized, how an incident should be routed, what approval is required, and how the response should be audited.

For business service intelligence, iDCOS converts technical findings into accountable operational processes.

Sensaka SmartBSM: service health, impact, and root-cause intelligence

SmartBSM organizes applications and infrastructure into business-service views. It supports service health, virtual-machine and business topology, relationship analysis, root-cause investigation, intelligent assistance, remediation, and business-impact visibility.

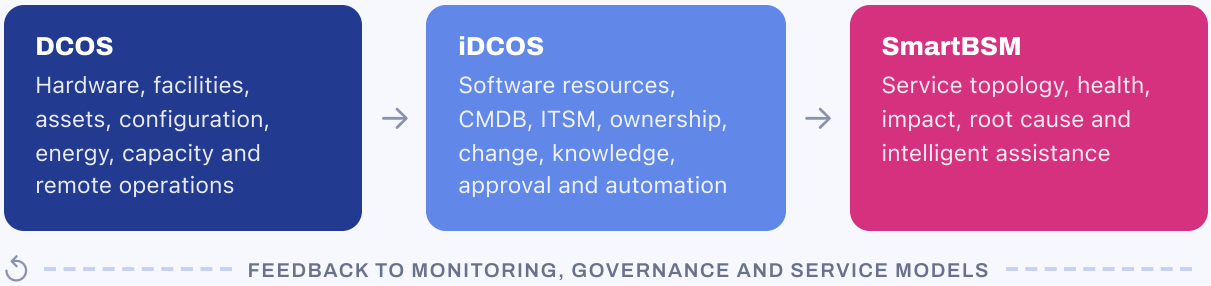
SmartBSM uses the service model to explain how abnormal conditions propagate across domains, identify credible initiating dependencies, distinguish technical severity from business priority, and help teams verify whether service has recovered.

For business service intelligence, SmartBSM provides the analytical and service-facing layer that connects operational evidence to business consequence.

Together, the three products create a closed operational loop:

1. **DCOS observes** physical infrastructure, components, facilities, energy, capacity, configuration, and change.
2. **iDCOS governs** resource identity, relationships, ownership, incidents, changes, knowledge, approvals, and controlled execution.
3. **SmartBSM understands** service health, dependency paths, impact, root-cause candidates, and business priority.
4. **Operational outcomes feed back** into event rules, service topology, knowledge, automation, lifecycle decisions, and risk controls.

FIGURE 9 — THE SENSAKA BUSINESS SERVICE INTELLIGENCE ARCHITECTURE



Sensaka connects physical evidence, operational governance, and service intelligence in one end-to-end operating model.

Conclusion: Observe Technology, Understand Services, Govern Outcomes

Infrastructure monitoring remains indispensable, but it is no longer sufficient for environments where business services span physical hardware, networks, storage, facilities, virtual platforms, cloud resources, software, and distributed operational teams.

The transition to business service intelligence begins by making technical evidence trustworthy, then connecting resources through a governed service graph. Normalized events, topology, configuration, change, ownership, and service objectives allow the organization to calculate meaningful health, identify impact, isolate credible root causes, prioritize response, and validate recovery.

The objective is not to conceal technical complexity behind a simplified dashboard. It is to make that complexity operationally understandable: engineers retain the evidence needed for diagnosis, service owners gain visibility into consequence, and management receives a defensible view of service risk and restoration.

Sensaka DCOS, iDCOS, and SmartBSM provide the complementary capabilities required for this progression—from component-level infrastructure observation to governed, service-aware operational intelligence.

Ten questions for a business service intelligence assessment

1. Can every critical service be linked to an accountable owner, service objective, and operating calendar?
2. Are physical, logical, cloud, software, and facilities dependencies represented in one governed relationship model?
3. Are resource identities reconciled across monitoring, CMDB, cloud, asset, and service-management systems?
4. Can repeated and downstream alerts be grouped without losing original diagnostic evidence?
5. Does service health account for degradation, redundancy, workload, capacity, and business period?
6. Can the organization distinguish confirmed impact from potential exposure?
7. Are root-cause candidates supported by topology, time, change, and metric evidence?
8. Do incidents automatically receive service, owner, configuration, change, maintenance, and knowledge context?
9. Is service recovery verified through business-facing indicators rather than device status alone?
10. Do resolved incidents improve service models, analytical rules, knowledge, automation, and risk controls?

If these questions require manual comparison across dashboards, documents, spreadsheets, and individual expertise, the organization has infrastructure monitoring—but not yet business service intelligence.

Learn more about Sensaka: sensaka.com
Request a business service intelligence assessment: sensaka.com/free-trial