OSPI DATA MODERNIZATION PROJECT PART 4: SKILLS GAP ANALYSIS FOR IT AND DATA TEAMS

2026

Deb Came, PhD
Assistant Superintendent of Data Analysis and Support

Ted Loran
Chief Information Officer

Prepared by:

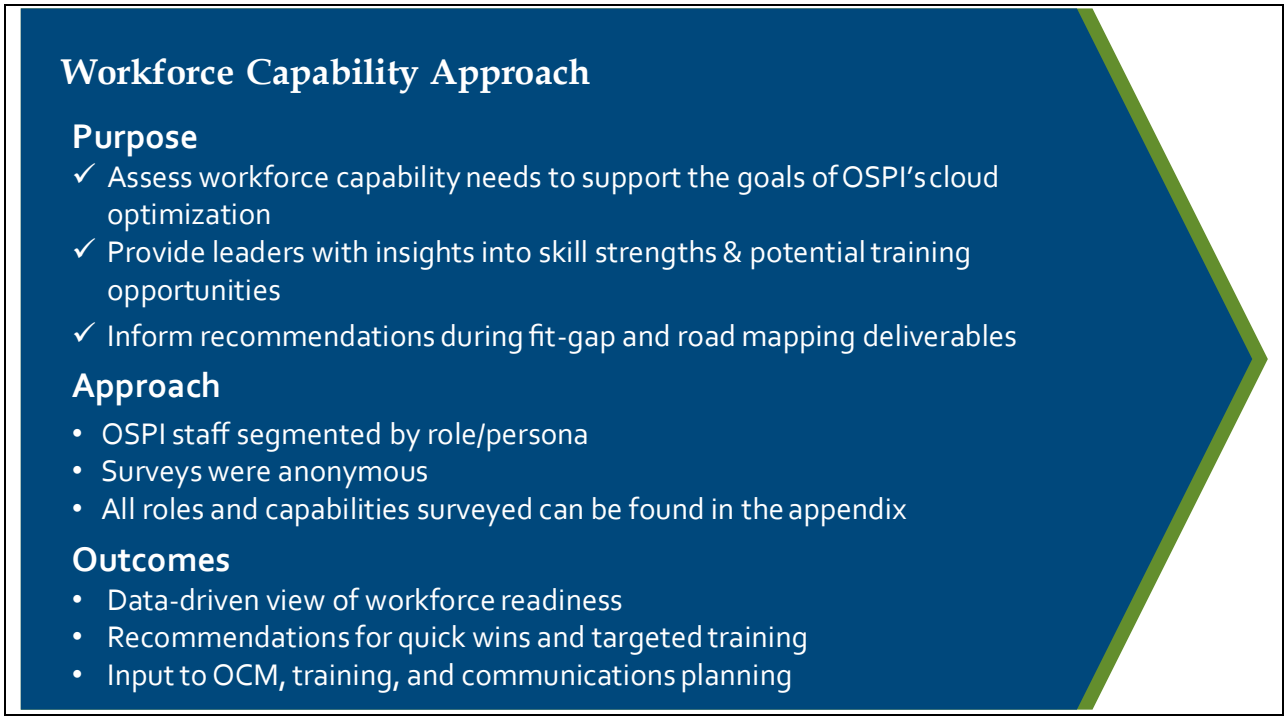
Kris Hicks-Green, Project Manager

Washington Office of Superintendent of
PUBLIC INSTRUCTION

Table of Contents

PART 1: SKILLS GAP ANALYSIS FOR IT AND DATA TEAMS

1.0 Workforce Capability Approach



2.0 Data by Persona

2.1 Workforce Capability Scores: Vision & Strategy Leadership

Workforce Capability Scores by Persona – 5 responses

Vision & Strategy Leadership				
Stakeholder Engagement/ Advocacy 5 | 0	Decision Making 4 | 1	Executive Leadership 4 | 1	Strategic Vision 4 | 1	Technical Strategy 3 | 2
Business Strategy 3 | 2	Risk Management/ Oversight 3 | 2	Communication 5 | 0	Stakeholder Influence 3 | 2	Change Management 3 | 2
Budget Approval/ Oversight 4 | 1	Vendor Oversight 4 | 1	Resource Allocation 4 | 1	Benefit Realization 2 | 3	Architecture Governance 1 | 4

Scores 4 and above | scores under 4

Average score	
5 – Deep Experience	3-3.9 - Some Experience
4-4.9 - SolidExperience	2-2.9 - Limited Experience
	1-1.9 - No Experience

Vision & Strategy Leadership Focus Areas

- **Strengths**: Solid experience in stakeholder engagement (4.0), communication (4.0), and budget/decision-making (3.6).
- **Gaps**: Architecture governance (2.3) and change management (2.8) show limited maturity.
- **Focus Areas**:
  - Build on strong engagement/communication by providing advanced leadership coaching.
  - Invest in structured change management frameworks to raise adoption capability.
  - Support architecture governance with targeted training or cross-agency mentorship.

Responses by Role:
1. Executive Sponsor (2)
2. Business Sponsor (3)

Total 5

2.2 Workforce Capability Scores: Architecture & Product Design

Workforce Capability Scores by Persona – 1 Response

Architecture & Product Design				
Enterprise application architecture 1 | 0	Agile Methodology 1 | 0	Solution Design 1 | 0	Stakeholder Alignment 0 | 1	TOGAF or Similar 0 | 1
API Management 1 | 0	Domain Architecture 1 | 0	Microservices Principles 1 | 0	Strategic Planning 0 | 1	Cross-functional Leadership 0 | 1
Cloud-native patterns 1 | 0	Integration Patterns 1 | 0	Roadmap Development 1 | 0	Governance Frameworks 0 | 1	

Average score	3-3.9 - Some Experience
5 – Deep Experience	2-2.9 - Limited Experience
4-4.9 - SolidExperience	1-1.9 - No Experience

Scores 4 and above | scores under 4

Architecture & Product Design Focus Areas

- **Strengths**: High scores in enterprise architecture (5.0), agile methodology (5.0), cloud-native patterns (5.0), and API management (5.0).
- **Opportunities**: Governance frameworks (2.0),TOGAF or similar (2.0), and cross-functional leadership (2.0).
- **Focus Areas**:
  - Formalize architecture governance through **industry certification training** or consider recruiting or contracting resources that have this experience.
  - Create **mentorship pathways** to strengthen cross-functional leadership.
  - Leverage **strong transferrable technicalskills** to accelerate roadmap execution.

Responses by Role:
1. Application Architect (1)

Total: 1

2.3 Workforce Capability Scores: Platform Development

Workforce Capability Scores by Persona – 5 Responses

Platform Development				
Environment Management 3 | 2	Scripting (Python/Powershell) 1 | 4	ETL Tools (E.G. Azure Data Factory) 1 | 3	Risk Management 1 | 4	Infrastructure As Code 0 | 5
Security Configuration 2 | 3	Source Control 1 | 4	CI/CD Integration 1 | 4	Advanced C# and .NET Core 1 | 4	Terraform or Bicep 0 | 5
Performance Optimization 2 | 3	Code Reviews 1 | 4	Security Integration 0 | 5	CI/CD Pipelines 1 | 4	Containerization (Docker Kubernetes) 0 | 5
Monitoring or Logging 3 | 2	Git Version Control 1 | 4	Agile Methodology 1 | 4	Debugging 1 | 4	Automation Frameworks 0 | 5
Monitoring And Alerting 3 | 2	Automation Via CI/CD 0 | 5	Microservices Architecture 1 | 4	Stakeholder Communication 2 | 3	REST API development 1 | 4
Release Planning 2 | 3	Pipeline Orchestration 2 | 3	C# and .NET Framework 1 | 4	Design Patterns 1 | 4	KPI Tracking 0 | 5
Average score: 3-3.9 - Some Experience	Azure Devops Or Jenkins 1 | 4	Agile Practices 1 | 4	Performance Tuning 2 | 3	Roadmap Alignment 0 | 5
5 – Deep Experience: 2-2.9 - Limited Experience	Product Lifecycle Management 2 | 3	Azure Networking 0 | 5	Mentorship 2 | 3	Unit Testing 1 | 4
4-4.9 - Solid Experience: 1-1.9 - No Experience				

Platform Development Focus Areas

- **Strengths**: Experience in performance optimization (3.4), environment management(3.4), and monitoring/logging (3.2).
- **Opportunities**: Terraform/Bicep (1.4), containerization (1.8), automation frameworks (1.8).
- **Focus Areas**:
  - Advance technical depth by focusing on skills development (ie. Terraform, Kubernetes, automation frameworks).
  - Potentially Recruit or contract for specialized infrastructure skills while building internal baseline.

Responses by Role:
1. Data Platform Engineer (1)
2. Cybersecurity Engineer (2)
3. Cloud Support Engineer (1)
4. Senior DevOps Engineer (1)

Total: 5

2.4 Workforce Capability Scores: Product Development

Workforce Capability Scores by Persona – 5 Responses

Product Development				
Mentorship 3 | 2	Stakeholder Communication 4 | 1	Monitoring or Logging 2 | 2	CI/CD Integration 0 | 5	Scripting (Python/Powershell) 1 | 4
Agile Practices 4 | 1	Design Patterns 4 | 1	Advanced C# and .NET Core 2 | 3	Security Integration 0 | 5	Roadmap Alignment 1 | 3
Debugging 3 | 2	C# and .NET Framework 3 | 2	Azure Networking 0 | 5	Performance Tuning 1 | 4	Monitoring And Alerting 1 | 4
Code Reviews 3 | 2	Git Version Control 2 | 3	Environment Management 0 | 5	REST API development 2 | 3	Terraform or Bicep 0 | 5
Agile Methodology 4 | 1	Unit Testing 3 | 2	Security Configuration 0 | 5	Microservices Architecture 1 | 4	ETL Tools (E.G. Azure Data Factory) 0 | 5
Risk Management 4 | 1	Performance Optimization 3 | 2	Automation Frameworks 0 | 5	Automation Via CI/CD 0 | 5	Containerization (Docker Kubernetes) 0 | 5
Release Planning 4 | 1	Product Lifecycle Management 3 | 2	Infrastructure As Code 2 | 3	Pipeline Orchestration 0 | 5	
Source Control 2 | 3	Azure Devops Or Jenkins 3 | 2	CI/CD Pipelines 0 | 5	KPI Tracking 0 | 5	

Scores 4 and above | scores und er 4

Average score

- 5 – Deep Experience
- 4-4.9 - Solid Experience
- 3-3.9 - Some Experience
- 2-2.9 - Limited Experience
- 1-1.9 - No Experience

Product Development Focus Areas

- **Strengths**: Experience in mentorship (4), risk management & release planning (3.8), stakeholder communication & agile practices (3.8) and source control (3.8).
- **Opportunities**: Terraform/Bicep (1.0), containerization (1.6), ETL tools (1.8), security integration (2.0).
- **Focus Areas**:
  - Advance technical depth by focusing on skills development (ie. Terraform, Kubernetes, ETL).
  - Lean into mentoring capacity to scale best practices across junior developer roles.

Responses by Role:
1. Product Manager (1)
2. Senior .NET Developer (1)
3. Junior .NET Developer (3)

Total: 5

2.5 Workforce Capability Scores: Data Development

Workforce Capability Scores by Persona – 24 Responses

Data Development				
Data Visualization 16 | 8	Data Quality Metrics And Monitoring Systems 12 | 12	Product Management Frameworks 4 | 19	Batch Processing 7 | 17	Go-to-market Strategy 3 | 21
SQL And Ml Concepts 15 | 9	Model Validation 8 | 16	Agile/Scrum 4 | 20	Cloud Storage Integration 2 | 22	Data Lake Architecture 0 | 15
Data Interoperability, Accessibility & Usability 12 | 11	Statistical Modeling 7 | 17	Predictive & Prescriptive Analytics 4 | 20	Python Or Scala 4 | 20	Automated Cataloging 1 | 23
Statistical Literacy 9 | 15	Data Governance Implementation 7 | 17	Experiment Design 6 | 18	Partitioning Strategies 1 | 23	Streaming Ingestion 0 | 24
Stakeholder Prioritization 9 | 15	Data Architecture With Infrastructure 5 | 19	Automation Design & Implementation 6 | 18	Python (Pandas Scikit - learn) 3 | 21	Apache Spark 0 | 24
Storytelling 9 | 15	Data Products Roadmap Planning 4 | 20	Data Product Economics And Democratization 3 | 21	Machine Learning Algorithms 2 | 22	Communication 14 | 9
Metrics Definition 9 | 15				

Average score

5 – Deep Experience	3-3.9 - Some Experience
4-4.9 - SolidExperience	2-2.9 - Limited Experience
	1-1.9 - No Experience

Scores 4 and above | scores under 4

Data Development Focus Areas

- **Strengths**: Some experience with SQL/ML concepts (3.2), data visualization (3.3), statistical literacy (3.0), stakeholder prioritization (3.0).
- **Gaps**: Streaming ingestion (1.4), Apache Spark (1.2), machine learning algorithms (1.6), Python/Scala (1.8).
- **Focus Areas**:
  - Target training in advanced analytics tools (Spark, Python/Scala) to strengthen foundation.
  - Enhance governance and interoperability practicesto support future scaling.
  - Leverage existing visualization and statistical literacy strengths to build data storytelling capacity.
  - Recruit or contract for specialized infrastructure skills while building internal baseline.

Responses by Role:

1. Data Analyst (15)
2. Data Product Manager (1)
3. Data Engineer (1)
4. 7 responses were unsure which role to select and were given a direct link to the Data Development Capability Survey

Total: 24

2.6 Workforce Capability Scores: Change Management

Workforce Capability Scores by Persona – 2 Responses

Change Management				
Viso or Lucidchart 2 | 0	Stakeholder Analysis 2 | 0	Continuous Improvement 1 | 1	Validation Testing 1 | 1	Impact Assessment 0 | 2
Program Planning 2 | 0	Communication 2 | 0	Documentation 1 | 1	Communication Planning 0 | 2	Workshop Facilitation 0 | 2
MS Project or Similar PM Tool 2 | 0	Risk & Issue Management 2 | 0	Requirements Elicitation 1 | 1	Gap Analysis 0 | 2	Change Frameworks 0 | 2
Budget Management 2 | 0	Stakeholder Management 2 | 0	Resource Allocation 1 | 1	User Story Creation 0 | 2	Resistance Management 0 | 2
Data Analysis 1 | 1	Process Mapping 2 | 0	Metrics Tracking 1 | 1	Training Development 0 | 2	BPMN Modeling 0 | 2
Average score 5 – Deep Experience 4-4.9 - SolidExperience	3-3.9 - Some Experience 2-2.9 - Limited Experience 1-1.9 - No Experience	Communication and Facilitation 0 | 2	Stakeholder Interviews 0 | 2	Scores 4 and above |scores under 4

Change Management Focus Areas

- **Strengths**: Stakeholder management, program planning, risk/issue management, and communication (all ~4.0).
- **Opportunities**: BPMN modeling (1.5), change frameworks (2.5), resistance management (2.5).
- **Focus Areas**:
  - Build structured process modeling capability to improve **documentation standardization**.
  - Standardize use of **change frameworks** (e.g., Prosci, ADKAR) to close adoption gaps and inform hiring.
  - Enhance resistance management and workshop facilitation skills through focused training, practice scenarios, or low-risk opportunities to apply them.

Responses by Role:
1. Program Manager (1)
2. Process Mapping Expert (1)

Total: 2

2.7 Rundown of Experience for Capabilities by Persona

Rundown of Experience for Capabilities by Persona

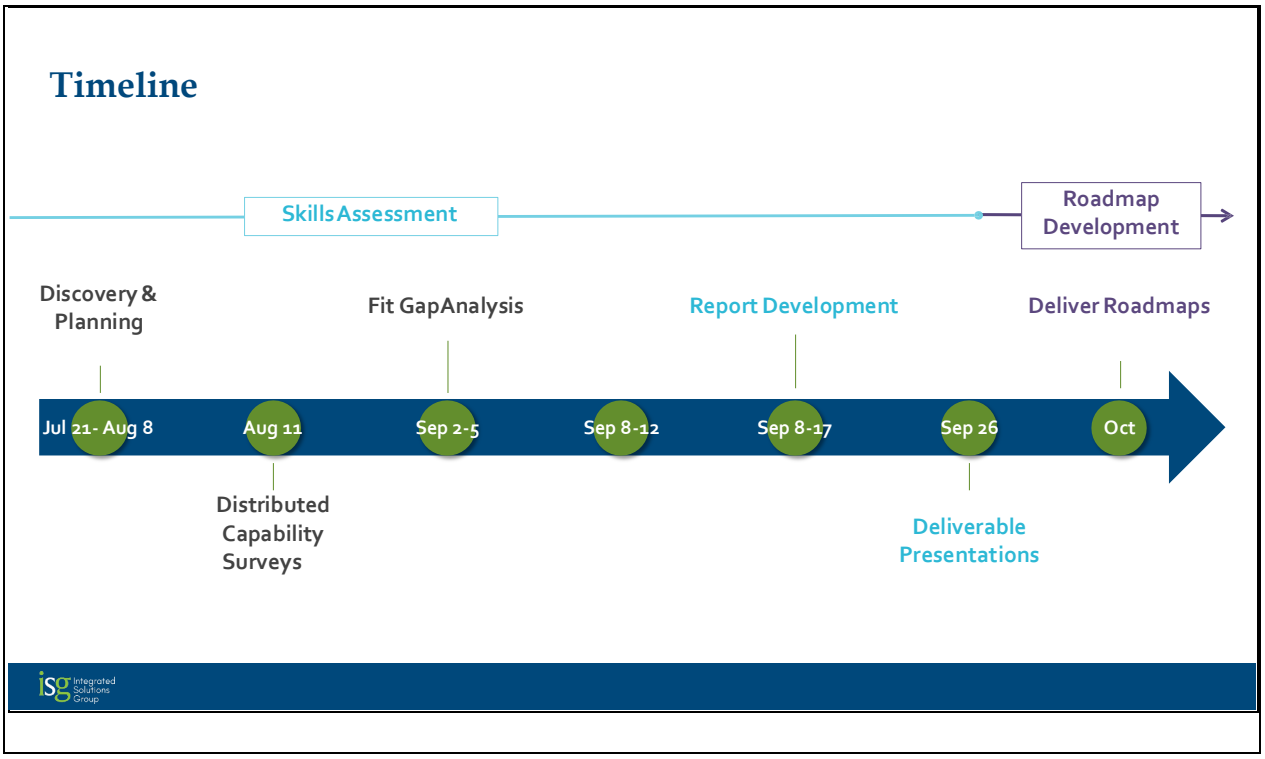
Persona Area	Deep XP	Solid XP	Some XP	Limited XP	No XP
Vision & Strategy	33	35	17	6	23
Architecture & Design	4	5	2	3	0
Platform Development	18	31	31	49	70
Product Development	17	52	52	47	29
Data Development	44	142	127	138	289
Organizational Change	1	24	24	4	1

- **Vision & Strategy**: Solid Experience (4.0)
- **Architecture & Data**: Some Experience (3.6)
- **Platform Development**: Limited Experience (2.2)
- **Product Development:** Some Experience (3.2)
- **Data Development:** Limited Experience (2.0)
- **OCM**: Some Experience (3.5)

Recommendations

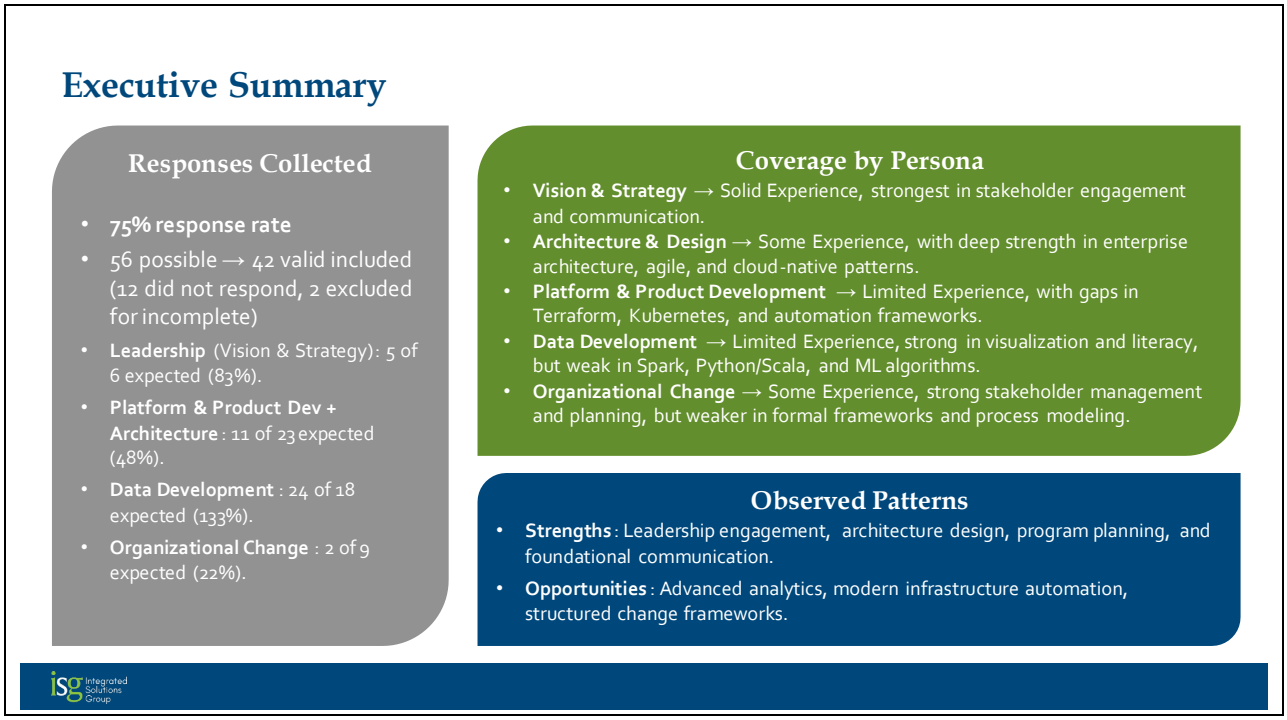
- Target internal training where foundations exist (OCM, agile practices, data visualization).
- Recruit or contract for specialized skills (advanced data engineering, containerization, DevOps automation) to close high-priority gaps in Platform and Data.
- Reinforce leadership capability with structured change management supports.

3.0 Next Steps



4.0 Appendix

4.1 Executive Summary



4.2 Breakdown of Findings

Rundown

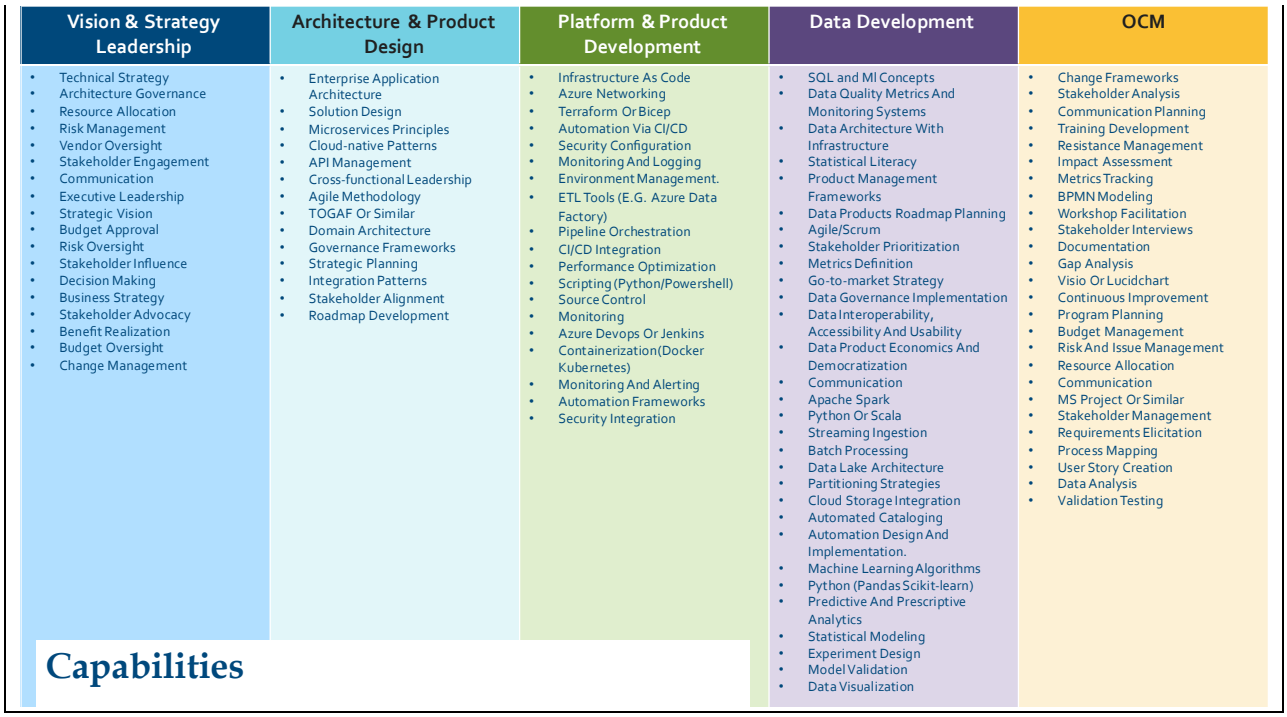
- **Vision & Strategy**: Scores indicate a solid foundation of leadership capability.
- **Architecture**: Some capability exists, but experience would ensure appropriate depth. With only one response in this area, it is hard to base much on the data.
- **Platform & Product Dev**: This is one of the two weakest areas and signals an immediate resourcing and development priority.
- **Data**: This is one of the two weakest areas and signals an immediate resourcing and development priority.
- **OCM**: Indicates strong in-house foundation — with additional training and exposure this capability can be sustained. However, there were no responses captured from individuals with the primary role of Organizational Change Manager.

4.3 Role Mapping

Role Mapping

Vision & Strategy Leadership	Architecture & Product Design	Platform & Product Development	Data Development	OCM
• **Executive Sponsor** • **Business Sponsor** • Technology Sponsor	• Enterprise Architects • **Application Architects** • Data Architects • Cybersecurity Architects	• Cloud Platform Engineers • **Data Platform Engineers** • **Senior DevOps Engineers** • **Cloud Support Engineers** • Junior Cloud Support Eng. • **Cybersecurity Engineers** • Test Automation Engineers • QA/Test Engineers • Product Business Analysts • **Product Managers** • **Senior .NET Developers** • **Junior .NET Developers** • Product QA/Test Engineers	• **Data Product Managers** • **Data Engineers** • BI Engineers • Data Scientists • **Data Analysts** • Data Business Analysts • Data QA/Test Engineers	• Change Managers • **Process Mapping Experts** • **Program Managers** • Project Managers • Business Analysts • Change Management Analyst/Trainers • Hypercare / Extended Time As-Needed Trainers

- We received responses from roles in **bold**.



Part 2: OSPI Current State Assessment

1.0 Current State Assessment

1.1 Overview

The Washington State Office of Superintendent of Public Instruction (OSPI) oversees K–12 public education across the state, providing leadership, resources, and accountability to ensure equitable educational opportunities for all students. OSPI works in collaboration with school districts, policymakers, and education partners to implement programs, manage funding, and deliver data and reporting that informs critical decision-making at both the state and local levels.

OSPI operates a complex data ecosystem spanning 138+ servers, 400+ database objects, and dozens of interconnected applications supporting Washington State's 1M+ students. While the analytical framework is quite sophisticated, critical organizational and governance gaps prevent reliable data-driven outcomes. Technology solutions can address approximately 75-80% of

current issues, but achieving 100% operational excellence requires fundamental organizational changes that no technology can solve.

This document shows a current state architecture baseline for OSPI's technology environment. It draws on available system information, stakeholder discussions, and practices found in shared documentation. The intent is to:

- Map current capabilities across infrastructure, applications, patterns, processes, change control, and governance.
- Assess maturity levels using a standardized heatmap to provide clarity on strengths and gaps.
- Elevate technology and data challenges that affect OSPI's ability to efficiently run and deliver on its mission.
- Provide actionable recommendations to guide modernization and improved alignment with best practices.

The goal is to guide OSPI toward a modernized future state that simplifies processes, improves data quality and accessibility, strengthens governance, and enables OSPI to support Washington's students, educators, and communities more effectively that simplifies processes, improves data quality and accessibility, strengthens governance, and enables OSPI to support Washington's students, educators, and communities more effectively.

This document provides an overview of the current state architecture. It is based on available documentation, stakeholder interviews, and observations. Where information is missing or incomplete, gaps are explicitly noted. This document should be treated as a baseline and updated iteratively as discovery continues.

1.2 Infrastructure

At present, OSPI's infrastructure environment supports a mix of core business applications, data services, and end-user tools. The technology landscape appears to be primarily hosted in traditional environments (on-premises and/or co-location facilities), with limited adoption of public cloud services to date.

Key infrastructure domains include:

- Hosting: Assumed to be predominantly on-premises or co-location, with potential hybrid extensions under consideration.
- Network Topology: Connectivity between agency offices, districts, and external partners is in place, though specific WAN, LAN, and VPN architectures have not been fully documented.

- Identity & Access Management: Active Directory is the foundational identity service, with possible integration into other application-specific IAM solutions.
- Security Controls**:** Firewall protections and endpoint security are in place, but detailed controls such as IDS/IPS, DLP, or advanced threat monitoring are not yet documented. Key controls, including role-based access controls (RBAC), conditional access, encryption policies, and key management are not defined. These gaps may affect OSPI's ability to implement zero trust principles, meet audit requirements, and maintain visibility as systems begin migrating to cloud environments.
- Business Continuity & Disaster Recovery: No documented disaster recovery or failover plans were shown. RPO/RTO objectives are not defined, and current systems do not appear to undergo regular backup validation or recovery testing.
- Infrastructure Provisioning and Management: No clear tooling strategy identified for infrastructure provisioning, monitoring, or automation (e.g., Azure DevOps, Terraform, etc.).
- Observability & Management: No clear strategy for centralized monitoring, alerting, and service management platforms (e.g., Splunk, Logic Monitor, ServiceNow, etc.) shown.

Gap Notes: Comprehensive diagrams and configuration documentation for hosting, networking, IAM, and security controls are not available currently. Further discovery will be needed to confirm the current state and identify specific modernization opportunities.

1.3 Applications

OSPI operates a comprehensive portfolio of 42+ web applications supporting statewide educational operations, reporting, and compliance requirements. Recent infrastructure discovery has revealed a substantial technology footprint built primarily on Microsoft .NET/IIS architecture with SQL Server databases. Undiscovered in the data are narratives of Based on interviews, there could be more Microsoft applications using Power Apps. Microsoft applications using Power Apps. Microsoft applications using Power Apps.

Discovered Application Landscape:

- Core Educational Systems: Educational Data Systems (EDS) with development (eds.dev.ospi.k12.wa.us), test and production environments supporting student information management and academic data workflows.
- Financial Management: Dedicated Financial Management System (FMS) with production (fms.ospi.k12.wa.us) and development instances managing budget tracking, fund

distribution, and financial reporting. Based on interviews, there is mention of Salesforce as the new Grant Management application.

- Reporting & Analytics: Report Card applications and ReportCardTableau systems providing reporting and data visualization capabilities for educational performance tracking.
- Specialized Applications: Domain-specific systems including Traffic Safety education management, Apportionment Services for fund distribution calculations, and various administrative consoles

Infrastructure Foundation:

- Technology Stack: Standardized on .NET/IIS applications (42+ instances) hosted on Windows Server 2022 infrastructure.
- Database Layer: 22+ SQL Server database instances with an average of 8+ user databases per instance, indicating significant data complexity.
- Server Infrastructure: 138+ total servers (122+ Windows, 16+ Linux) supporting the application portfolio.
- Environment Management: Clear separation of production, development, and UAT environments across key systems.

Key considerations include:

- Application Proliferation: 15 "Default Web Site" instances suggest potential consolidation opportunities or framework improvement.
- Integration Patterns: Direct database connections appear prevalent with limited middleware or service layer abstraction.
- Documentation Gaps: While infrastructure is discoverable, application dependencies, data flows, and integration mappings require comprehensive documentation.
- Deployment Maturity: Infrastructure suggests traditional deployment approaches using TFS and beginnings of Azure DevOps with opportunities for CI/CD and DevSecOps modernization.

Key Application Systems (in discovery data):

Report Card & Analytics Performance reporting and data visualization • ReportCard platform	**Educational Data systems** Student information and academic data management

• ReportCardTableau integration • Educational outcome analytics • ReportCard platform	• EDS environments (prod, test, dev) • Main interface for school districts • Student data workflows
Financial Management Budget tracking (potentially legacy system that still tracks data) • FMS application • Fund distribution • Financial reporting	**Specialized Applications** Domain-specific and administrative systems • Traffic Safety education • Specific services • Administrative tools (e.g. consoles)

Web Application Portfolio Distribution

Core Business Systems • EDS, CEDARS, FMS, Apportionment • Mission-critical applications • Default IIS sites and standard hosting • Common web application foundations *50% of all apps*	**Specialized Applications** • Domain-specific business applications • Custom-built education solutions *26% of all apps*
Specific Services • WINS, WSUS, PKI, Exchange • Network and security tools *12% of all apps*	**Administrative Consoles** • Management interfaces • System administration tools *7% of all apps*
Report Card & Analytics • Report Card and visualization tools' • Internal and external dashboards *5% of all apps*	

Server Infrastructure Distribution

Windows Server • Windows Server 2022 Standard • Primary application hosting environment *88% of all apps*	**Linux Infrastructure** • SUSE Linux Enterprise • Analytics and specialized workloads *12% of all apps*

Strategic Considerations: The discovered portfolio represents a mature but complex environment ready for systematic modernization. The standardized Microsoft technology stack provides a clear migration path to Azure Cloud, while the large database footprint indicates opportunities for consolidation and optimization.

1.4 Tools Inventory

OSPI's current tooling environment reflects a mix of traditional on-premises utilities, Microsoft platforms, and limited public cloud adoption. While some enterprise tools like Power BI and Power Apps have emerged in pockets, the broader OSPI tools ecosystem lacks centralized oversight, tool standardization, or a defined tooling strategy. The following table outlines known tools and platforms in use as discovered during the infrastructure and application review:

Discovered Tooling Landscape:

Category	Tool	Description
Cloud Platforms	Microsoft Power Platform	Primary cloud platform; used for app hosting and analytics (e.g., Power Apps, Power BI)
	Microsoft Fabric	IT and Data teams are using tools in Azure like Fabric to explore better analytics capabilities; no wide adoption noted, Child Nutrition team has an instance of Fabric in production-level, HR is working on standing up something for production.
Applications	.NET / C# / ASP.NET	On-prem application suite under Microsoft that runs about 60% of all OSPI apps interfaced through EDS.
	JavaScript	Used for frontend development in ICOS to enable user interactions like filtering and downloading reports in PDF, Excel, or Word formats.
	GIS (Geographic Information System)	In-house development integrated with ICOS for pre-disaster modules and mapping building-level data to activities and student impacts.
Data Storage & Integration	SQL Server / SSIS	Widely used for legacy ETL and data warehousing tasks.
	SLDS (Statewide Longitudinal Data System)	Past longitudinal data project (about 10 years ago) that failed due to lack of organizational buy-in and developer scheduling issues; remnants in data marts.
	Redgate SQL Monitor	Basic capabilities used for monitoring SQL Server performance; limited usefulness for analyzing performance.
	Access Databases	Still in use in certain departments for legacy reporting.

Category	Tool	Description
	Excel	Extremely prevalent across departments; used for data entry, validation, reporting, and reconciliation.
	PowerShell / Manual Scripting	Used in data transformations such as moving vendor sFTP files, and other file movement management.
	CSV / Flat Files via sFTP	Common for cross-system data exchange between districts and OSPI.
	API	Employed for one program's nightly data handoff to external testing systems, facilitating demographic information outflow.
	C# / C++ Jobs	Nightly automated scripts that pull data from sFTP sites or send to external sites, separate from SQL Server execution.
	Stored Procedures	Dictate data processing and business rules in databases, updated annually for new job information and federal policy compliance.
	Washington Query / Query 2006	Core tool for ad-hoc data pulls and federal reporting (e.g., NCLB fact tables), with SPSS syntax for downloads.
	SharePoint	Primary platform for sharing data files (e.g., CSVs from Student Information) and collaborative access across teams.
	System Drives (e.g., S-drive)	Legacy storage folders for file management.
Reporting & Visualization Tools	Tableau Server (not SaaS version)	Used widely for analytics, especially for public-facing analytics tools like Report Card.
	SSRS (SQL Server Reporting Services)	Used for standard reports and document storage.
	R	Programming language used by a few analysts for deeper data analysis; siloed to a few analysts and a couple of teams.
	Python	Utilized for data analysis within the ICOS console to evaluate efficacy, such as correlating HVAC scores with report card grades and enrollment.

Category	Tool	Description
	Power BI	Deployed in Power Platform for analytics and potentially small instances inside Fabric, but not yet mature in self-service usage. Usage varies by department.
	Excel	Serves as a fallback visualization and reporting tool for users not running SQL scripts.
Student Education Systems	CEDARS / CEDARS Loader	OSPI's centralized education data repository system where districts submit student information; heavily customized.
	IEP Systems	Multiple solutions used across LEAs to submit special education data; not standardized statewide.
	School Apportionment System (SAFS)	Custom internal tools for financial data submissions and apportionment funding calculations; aging architecture with heavy manual processes.
Identity, Access, & Security	EDS	Main legacy custom-built authentication tool that hosts all OSPI's applications and central access for districts.
	Entra ID	Sparsely being used for applications outside of EDS; no current adoption strategy, WaTech compliance waiver ends September 2025.
	CommVault	Used for daily backups on all servers; no data purging and data continues to accumulate year-after-year.
Productivity, Collaboration, and Project Tools	Microsoft Teams / Outlook / Office 365	Primary communication and productivity suite.
	Microsoft Forms	Used for surveys internally and externally for ad-hoc projects.
	Smartsheet	Used for some transitioned iGrants activities that were not moved to Salesforce Grantmaking.
	Visio	Used for diagramming technical processes.
	Miro	Used for interagency collaboration "whiteboard" and project management visuals.
	TFS (Team Foundation Server)	Legacy tool being used for application development lifecycle, also used for IT ticketing, data pipeline tasks, and

Category	Tool	Description
		business rule implementation; currently planning a transition over to Azure DevOps.
	Azure DevOps	Database development teams have transitioned from TFS into Azure DevOps.
	File Maker	Internal database for processing and running findings from EGMS data, supporting compliance reviews and non-compliance summaries. Legacy tool that is phased out with adoption of EGMS.
	iGrants	Legacy grants management system replaced by EGMS; previously used for special education funding and reporting. Some historical data may show up from iGrants.
SaaS / Vendor Tools for Public-Facing Program Management	Salesforce Grantmaking	Currently called EGMS (Electronic Grants Management System) and replacing legacy iGrants system, used for federal reporting indicators, but clunky navigation and high learning curve, also lacking File Maker capabilities.
	Alchemer	Survey software for data collection (e.g., Capstone facilities reports, program grants); enables flexible question adjustments but lacks real-time validation.
	eVal	Vendor-managed tool for school districts evaluations.

1.5 Standards

OSPI has varying standard documented procedures for how its technology systems are built and connected. Integration practices, architecture practices, and master data practices can vary based on which team did the work. Different teams follow their own rules for naming files, organizing data, and storing documents, which creates a lack of clarity and inconsistencies across the organization.

Key considerations include:

- Integration Style: Integration relies heavily on batch processes and point-to-point connections, with limited evidence of standardized API-first or event-driven approaches.
- Data Architecture Fragmentation: There is no formal enterprise data architecture in place. Systems work independently, and no centralized or federated data access model exists to support cross-system reporting or analysis.

- Metadata & Master Data: OSPI lacks a unified metadata layer or master data strategy. This hinders data lineage, consistency, and trusted reporting across domains such as student data, finance, and HR.
- Federation and Cross-Agency Access: Current patterns do not support scalable data federation across internal or partner agencies, limiting the ability to share data securely and efficiently for statewide decision-making.

Gap Notes: Reference architectures and repeatable solution patterns are not currently defined or enforced.

1.6 Processes

Technology processes across the agency support day-to-day delivery but lack formal standardization and automation.

Key considerations include:

- Coding Standards: No consistent coding or architectural standards are confirmed to be in place.
- Infrastructure Patterns: The environment is characterized entirely by monolithic applications deployed on traditional infrastructure. There is no current adoption of containers, or serverless patterns across the portfolio.
- SDLC / DevOps: Development life cycle approaches vary. Evidence suggests limited CI/CD pipeline adoption, with manual deployments still in use.
- Incident & Problem Management: Issues are addressed reactively, but no clear ITIL-aligned process has been confirmed.
- Testing Approach: Testing is assumed to be primarily manual, with no consistent automation framework in place.

Gap Notes: Standardized and automated processes (e.g., CI/CD, ITIL-aligned service management, test automation) require definition and rollout.

1.7 Workforce

The internal workforce at OSPI demonstrates commitment and institutional knowledge, but lacks exposure to many of the modern skills, roles, and tools needed to support a cloud modernization at scale. There are a few key considerations that are widespread across state government:

- Limited staff time for training: Employees juggling with full workloads, leaving little time to participate in training or skill-building.
- Lack of role clarity: Ambiguity around responsibilities and decision-making creates duplication, missed tasks, and reduced accountability.
- Budget limitations for hiring or training spending: Limited resources constrain hiring specialized talent and funding training, creating persistent skill gaps.
- Resistance to change and change fatigue: Frequent initiatives and shifting priorities leave staff weary, skeptical, and less willing to adopt new approaches.

Key considerations include:

- Skills Coverage: Current staff roles do not appear to include dedicated Cloud Architects, DevOps Engineers, Platform Engineers, DataOps practitioners, or FinOps analysts -- positions that are typically critical in cloud-first environments.
    - The Platform Development persona show limited experience on average, with Terraform/Bicep (1.4), containerization (1.8), and automation frameworks (1.8) all critical gaps. See appendix 1 for a reference to persona based capabilities.
- CI/CD and Automation: There is limited experience with continuous integration/continuous deployment (CI/CD) pipelines, Infrastructure as Code (IaC), and automation tools, which presents a barrier to efficient, repeatable delivery.
- Cloud Expertise: While some exposure to cloud concepts exists, hands-on familiarity with Azure or other public cloud environments is minimal. Most staff are grounded in on-premises operations.
    - Only one Architecture representative responded and reported strong cloud-native skills (5.0), while governance and The Open Group Architecture Framework (TOGAF) knowledge scored just 2.0. See appendix 1 for a reference to persona based capabilities.
- Training and Upskilling: No structured upskilling or cloud enablement programs are in place. Staff development appears informal and ad hoc, without a competency framework tied to modernization goals.
- Partner Augmentation Needs: Given the scope of cloud transformation, OSPI may need to augment with external partners in the near term to provide architectural, engineering, and operational support.

- Organizational Structure: OSPI's organizational structure reflects strong leadership and engagement, though opportunities remain to strengthen balance by clarifying roles and augmenting capacity in technical and OCM functions.
    - Vision & Strategy (83% response) and Data (133%) were well represented, however, Change Management saw only 22% of expected responses. See appendix 1 for a reference to persona based capabilities.

Gap Notes: OSPI will need a structured workforce development plan that defines required future-state roles, maps existing skills, and addresses gaps through training, hiring, or vendor support. Without this, modernization efforts will rely too heavily on unskilled or overallocated resources, slowing long-term sustainability.

1.8 Change Management

Change management practices are not fully documented and vary across teams, which creates risk to stability and consistency. In our discovery stage, we did not have involvement from dedicated change staff. Without this established practice, OSPI risks inconsistency in adoption and resistance management.

Key considerations include:

- Approval Process: Approvals may occur informally or through ad-hoc management review, conversations, and/or email, without a formal Change Advisory Board (CAB).
- Tooling: Systems such as ServiceNow or Jira are not used for ticketing and formalized workflows are not established.
- Rollback Procedures: Rollback and emergency change management processes are not documented.
- Low Formalized Change Capability with BPMN modeling (1.5), resistance management (2.5), and change frameworks (2.5) all underdeveloped. See appendix 1 for a reference to persona based capabilities.

Gap Notes: A formalized, standardized change control process with governance, workflows, and rollback procedures is not yet in place.

1.9 Change Control

Current change control practices within the organization are inconsistently applied and lack a standardized framework, leading to heightened risk around system stability, auditability, and operational consistency. While individual teams have developed their own informal approaches,

these methods vary significantly and are often undocumented, making it difficult to ensure accountability or repeatability. Without a unified governance model, changes may be implemented without sufficient oversight, introducing the potential for service disruptions, security vulnerabilities, or compliance gaps.

Key considerations include:

- Approval Process: Approvals may occur informally or through ad-hoc management review, conversations, and/or email, without a formal Change Advisory Board (CAB). Tooling: Systems such as ServiceNow or Jira are not used for ticketing and formalized workflows are not established.
- Rollback Procedures: Rollback and emergency change management processes are not documented.

Gap Notes: A formalized, standardized change control process with governance, workflows, and rollback procedures is not yet in place.

1.10 Communications

OSPI staff consistently report uncertainty about their roles in the modernization effort and how upcoming changes will impact day-to-day responsibilities. Organizational readiness remains limited by unclear communication channels and the absence of a structured change network. Without targeted communication and embedded champions, staff risk feeling disconnected from the modernization journey, which may slow adoption, reinforce resistance, or increase turnover.

Key considerations include:

- **Role Clarity:** Many staff do not understand how modernization affects their specific responsibilities, creating confusion and hesitation to engage.
- **Communication Channels:** Current updates are ad hoc and vary by team; no agency-wide cadence exists for progress reporting or feedback loops.
- **Message Framing:** Staff often perceive change as disruptive rather than supportive, with limited emphasis on "what's in it for me".
- **Change Network:** There is no structured group of sponsors, leads, and champions to localize messages, surface risks, or model adoption behaviors.

Gap Notes: Without structured communications and a change network, modernization will rely heavily on informal channels, leading to inconsistent understanding and uneven adoption across teams.

1.11 Infrastructure Governance

Governance structures are limited and inconsistently applied, resulting in fragmented practices across the organization. Policies and standards, such as naming conventions, tagging, cost management, and security, are not consistently defined or enforced, leaving critical areas open to interpretation. While IT governance appears centralized, the degree of departmental ownership is unclear, creating uncertainty in decision-making and accountability. Without formal enterprise-wide bodies, such as an Architecture Review Board or Data Governance Council, the organization faces risks of policy drift, inconsistent oversight, and reduced alignment with modernization and compliance goals.

Key considerations include:

- Policies: Naming conventions, tagging standards, cost management policies, and security standards are not consistently defined or enforced.
- Organizational Model: IT governance appears centralized, though the degree of federated ownership by departments is unclear.

Gap Notes: Enterprise-wide governance processes and bodies (e.g., Architecture Review Board, Data Governance Council) need to be formally established.

1.12 Data Governance

Data governance at OSPI lacks both automated systems and adequate staffing. The data governance committee does not have enough members or authority to enforce data standards and policies.

Key considerations include:

- Governance Tools: OSPI has no automated tools to manage data security, quality checks, or compliance monitoring. All governance activities must be done manually, creating significant workload and increasing risk of human error.
- Policy Awareness: While data governance policies exist in written form, most staff members are either unaware of these policies or do not consistently follow them in their daily work.
- Roles and Accountability: The organization has not formally assigned data stewards or data owners across the agency. Documented data governance roles and responsibilities are lacking for data being democratized internally and externally, leading to unclear decision-making authority and data quality issues.

Gap Notes: OSPI lacks a comprehensive data governance framework with automated enforcement tools, and organization-wide adoption of data management standards. Current governance efforts are manual, understaffed, and lack the authority needed for effective oversight.

1.13 Data Architecture & Management

Core Data Systems and Infrastructure

OSPI operates a complex data ecosystem that includes several primary data systems:

CEDARS (Comprehensive Education Data and Research System)

- Functions as the central student information database collecting enrollment, demographic, and program participation data.
- Receives data from district SIS systems via file uploads with specific formatting requirements.
- Uses a "logical delete" validation process where entire submissions are rejected if error threshold exceeded.
- Nightly processing cycle with 13 SQL Agent Jobs running sequentially - single failure can halt entire pipeline.
- Integration with multiple downstream systems including assessment and reporting
- Processes data for approximately 1.1 million students across 295 districts

EDS (Education Data System)

- Core authentication platform and interface for school districts.
- Houses 42+ applications and modules with varying security models.
- Currently under WaTech waiver for not being in Entra ID - significant security compliance gap.
- Single point of access for majority of district-facing applications.

SAFS (School Apportionment and Financial Services)

- Manages critical financial processes including district budgets (F-195), annual financial statements (F-196), initial budget planning (F-203), and personnel data reporting (S-275).
- Calculates and distributes almost $30 billion in state funding to school districts through the apportionment system.
- Relies on manual file uploads and batch processing for all financial data collection.

- Financial data is manually reconciled across multiple applications.
- Districts submit data through separate forms for each financial process.
- Manual validation is done regularly between related financial forms (F-195, F-196, F-203).

eCert (Educator Certification)

- Manages educator certification records and processes for approximately 75,000 active educators in Washington State.
- Processes initial certifications, renewals, endorsement additions, and interstate reciprocity for teachers, administrators, and educational staff associates.
- Maintains certification status data that determines educator eligibility to work in specific roles and subject areas within school districts.
- Exchanges data with the S-275 Personnel Reporting system and other OSPI educator databases through file-based transfers.
- Operates using file export and import processes for data exchange with district systems and other OSPI applications, without direct API connectivity.

Data Architecture & Integration

Current Architecture State:

- Heavily siloed systems with limited integration capabilities
- Manual data transfers common between systems - estimated 60% of data movement is manual.
- Multiple separate SQL Server databases - discovered 400+ databases across 139 servers.
- No centralized data warehouse architecture - each system keeps its own version of truth.
- Staff estimate 30% of their time is spent on data movement rather than analysis.

Integration Methods:

- SFTP file transfers from districts and vendors
- Manual CSV exports/imports
- Manual surveys and forms through SaaS or EDS applications
- Nightly batch processing to copy data from web app servers to database servers for analytics

Data Validation:

- Logical Delete process in CEDARS rejects entire submissions for threshold violations.

- Manual verification steps between each system handoff
- Limited automated validation - primarily basic format checking
- Districts report resubmitting data 3-4 times on average due to validation failures.
- Districts discover errors only after full submission, requiring complete reprocessing.

Technical Infrastructure

Database Systems:

- SQL Server on-premises (versions ranging from 2008 to 2019)
- Multiple separate databases per domain with no consistent naming conventions
- Limited cloud adoption - only pilot projects in cloud
- Databases designed for thousands of records now have millions.
- Inconsistent backup schedules across systems, recovery time objectives (RTO) and recovery point objectives (RPO) undefined.

Development Environment:

- .NET Framework 2.0-4.5 (many versions past end-of-life)
- Custom Windows Services without standardized logging or monitoring
- Outdated frameworks have known vulnerabilities.
- Different technology stacks require specialized knowledge, creating key person dependencies.

Reporting Tools:

- Most visuals are on Tableau with minimal data modeling before being stored in Tableau Server
- Excel for data manipulation and completely disconnected from databases
- R code for some analysis maintained by individual analysts
- Manual SQL queries with limited version control or documentation
- Tableau queries against internal Tableau Server storage affect transaction processing and total crashes, which results in rebuilding of Tableau Server to continue work
- No consistent calculation definitions automation or enforcement across reporting tools

Additional Infrastructure Observations:

- Unclear disaster recovery site for critical systems

- Authentication mechanisms vary by application (Windows Auth, custom EDS auth, potentially hard-coded credentials found in some legacy systems)
- Network segmentation insufficient - discovered unclear pathing between test and dev and production environments
- Monitoring limited to basic server health - limited application performance monitoring
- Change management process exists but frequently bypassed for "emergency" fixes

Scale and Constraints

Current Environment Statistics:

The Azure Migrate discovery captured only a fraction of OSPI's true data ecosystem due to not having installed the Azure Migrate agent on production systems (*this decision was made to mitigate any potential risk to those systems during a Discovery project phase*). Without direct access to the current web applications, servers, and databases, the discovery data reflects a partial view of OSPI's actual technology footprint. Despite this limitation, the patterns revealed provide critical insights into the scale and complexity of OSPI's data environment.

Metric	Discovery Data	Estimated Actual	Implication
Allocated Disk Space	122 TB	200+ TB	Significant undiscovered data stores
Web Applications	43	60+	Many internal and external portals unmapped
Data Usage	60 TB	100+ TB	Actual data far exceeds discovery
Servers	139	150+	Infrastructure complexity understated

Discovery Data Limitations:

- Majority of business processes have infrastructure gaps that cannot be gathered narratively.
- Most web apps/portals are missing from discovery data.
- Critical production systems missing from discovery data or not clearly named.
- Backup infrastructure cannot be analyzed from discovery data.

1.14 Top 20 Most Frequent Usage Data Terms:

This data is not comprehensive, but from a quick overview, "report card" occurs the most often, which means it has the highest complexity compared to the other usage types.

#	Occurrences in the discovery data	Term
1	49x	report card
2	33x	data requests
3	17x	data analysis
4	7x	tableau educator dashboard
5	6x	data analysis requests
6	3x	public records requests
7	3x	report card data source for rc_rb_teacherratioprogramcertificate
8	3x	retentionmobility
9	2x	data source for rc_rb_teacherratioprogramcertificate for report card
10	2x	rc_rb_teacherqualificationsummary for report card
11	2x	computer science workbook
12	2x	certification history look up tool
13	2x	slds grant
14	2x	data source for building rc_teacherdemographics for report card
15	2x	one dashboards
16	2x	teacher quality comparison workbook
17	2x	data source for rc_rb_studentaccesscontent for report card
18	2x	eds applications
19	2x	sees survey
20	2x	teacherlmcotfdistrictsubmission

The Hidden Complexity

The discovery gaps reveal a critical constraint:

OSPI lacks comprehensive visibility into its own data ecosystem. Essential systems operate in isolation, known only to specific teams. Mission-critical processes like educator certification, special education services, and federal program management exist outside the discovered infrastructure, suggesting parallel shadow IT systems evolved to meet immediate and growing organizational needs.

Data Fragmentation Indicators

Analysis of usage patterns reveals the true complexity. "Report Card" appears 49+ times across different systems - not as a single system, but as fragments scattered across databases, each having partial contributions to the full breadth of all Report Card calculations. This pattern repeats across domains:

- Student data exists in enrollment systems, assessment platforms, and program databases - each with different processes and handling
- Educator information spans certification, assignment, and qualification systems – and does not seem to be centralized like other domains (e.g. CEDARS, EDS)
- Financial data flows through multiple calculation engines - each applying different business rules, then exits and is managed manually with excel

Operational Constraints

The scale creates compound constraints:

Overwhelming Volume: With 100+ TB of actual data across hundreds of systems, comprehensive analysis becomes impossible without automation. Manual processes that worked at smaller scales now consume entire work weeks.

Complexity Paralysis: The interconnected nature of 400+ database objects means changes in one system cascade unpredictably. Teams are weary of improvements, knowing that fixing one issue might break five others.

Knowledge Fragmentation: No single person understands the complete data landscape. Expertise is siloed by system, creating dependencies on specific individuals who hold pieces of the larger puzzle.

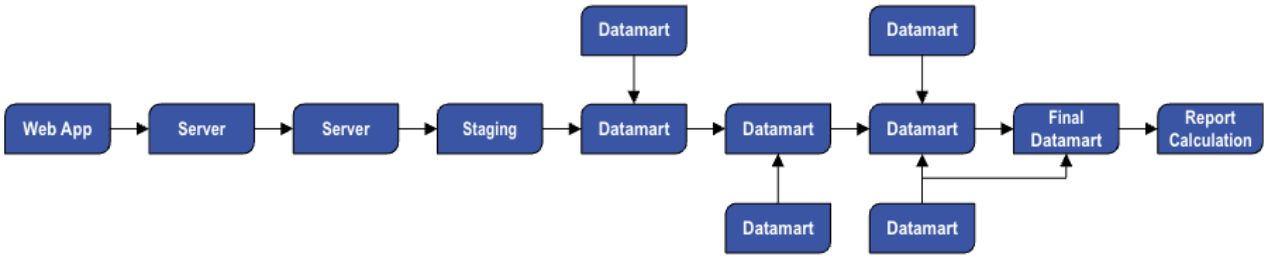
Performance Degradation: Systems designed for thousands of records now process millions. What worked reliably for decades now strains under exponential data growth and increasing analytical demands.

The Multiplication Effect

The discovery data hints at a larger pattern - every visible system has 2-3 supporting databases, staging areas, and backup structures; not to mention spread across Prod / Test / Dev environments. A single "Report Card" query might traverse:

- Multiple source databases for raw data
- Staging databases for transformation
- Intermediate calculation stores
- Final presentation databases

Process for one Report component:



This multiplication means that improving any single metric requires understanding and modifying multiple systems, each with its own logic, ownership, and constraints.

Strategic Impact

These constraints aren’t just slow operations - they fundamentally limit OSPI's ability to answer critical questions:

- Which interventions improve student outcomes?
- How do a range of factors interact to influence graduation rates?
- Where should limited resources be invested for maximum impact?

Without the ability to connect data across domains efficiently, OSPI operates on incomplete pictures, making decisions based on fragments rather than comprehensive insights.

The scale isn't just about size - it's about complexity that has grown beyond human ability to fully understand or manage without modern architectural approaches.

1.15 Data Operations & Process Obstacles

Current Pain Points and Limitations

Initial feedback has shown challenges such as data silos, slow data request response times, and maintenance issues with aging infrastructure. While these systems have served OSPI reliably for decades, our analysis reveals that what appears as stability masks deeper structural challenges that limit OSPI's ability to meet modern educational data needs.

Critical Operational Impacts: The current environment requires extensive manual reconciliation between systems, with staff spending considerable time validating data across multiple sources to ensure accuracy. What should be straightforward data requests often require days of investigation to find authoritative sources. Federal compliance reporting, while successfully completed, demands significant manual effort and coordination across teams. The absence of automated validation means quality issues are discovered late in processes, requiring reactive remediation rather than proactive prevention.

Systemic Technical Debt: OSPI maintains numerous databases across multiple systems, each evolved to meet specific needs over time. While individually functional, these systems lack integration, resulting in the same data being calculated and stored differently across departments. Critical business logic exists in stored procedures and custom code written by staff no longer with the organization, creating knowledge gaps. The architecture, built incrementally over decades, makes comprehensive changes challenging without risking disruption to working processes.

Human Resource Constraints: The technical complexity requires specialized knowledge that has accumulated in long-tenured staff members. This expertise, while valuable, creates vulnerabilities when key personnel are unavailable or retire. The absence of self-service capabilities means analytical staff serve as intermediaries for routine data requests, limiting their capacity for strategic analysis. Training new staff requires extensive mentorship periods due to undocumented business rules and system interdependencies.

Data Quality and Governance Gaps: Without centralized governance, departments have developed their own definitions and validation rules, leading to inconsistent interpretations of the same metrics. The lack of transparent calculation documentation makes it difficult to explain how specific numbers are derived, particularly for complex federal formulas. When discrepancies arise between reports, finding the root cause requires manual investigation through multiple systems. Historical reproducibility is limited, making year-over-year comparisons and audit responses time-intensive.

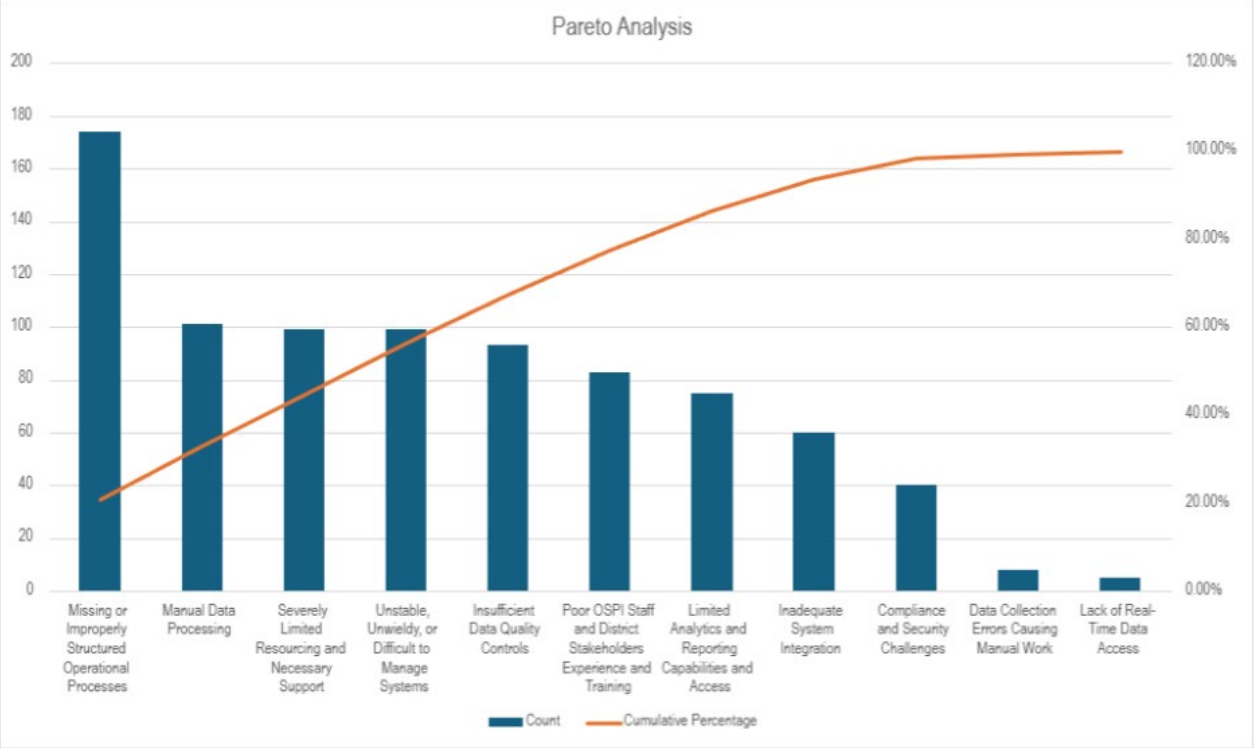
District and School Impacts: School districts navigate multiple submission systems with varying requirements and timelines. While these systems function, they require districts to maintain

expertise in numerous platforms and data formats. Validation occurs post-submission, meaning districts may not discover issues until after critical deadlines. Smaller districts with limited technical staff face proportionally greater burdens in meeting complex submission requirements across multiple systems.

Strategic Limitations: The current infrastructure, while operational, constrains OSPI's ability to provide prompt, comprehensive insights for policy decisions. Cross-domain analysis requires manual data compilation from multiple systems, limiting the ability to understand interconnected factors affecting student outcomes. The absence of predictive capabilities means interventions are reactive rather than preventive. Modern expectations for real-time data access and self-service analytics cannot be met within current architectural constraints.

Data Issue Analysis

During the Discovery Phase, over 800 issues were documented across OSPI collectively. Analysis shows that solving the first 6 issue categories would address over 77% of OSPI's challenges:



This analysis shows that if we solve the first 6 Issues Categories of:

- Missing or Improperly Structured Operational Processes
- Manual Data Processing
- Severely Limited Resourcing and Necessary Support

- Unstable, Unwieldy, or Difficult to Manage Systems
- Insufficient Data Quality Controls
- Poor User Experience and Training for OSPI Staff and District Stakeholders

It would solve over 77% of OSPI's issues. With the number one priority being "Operational Processes Missing or Improperly Structured."

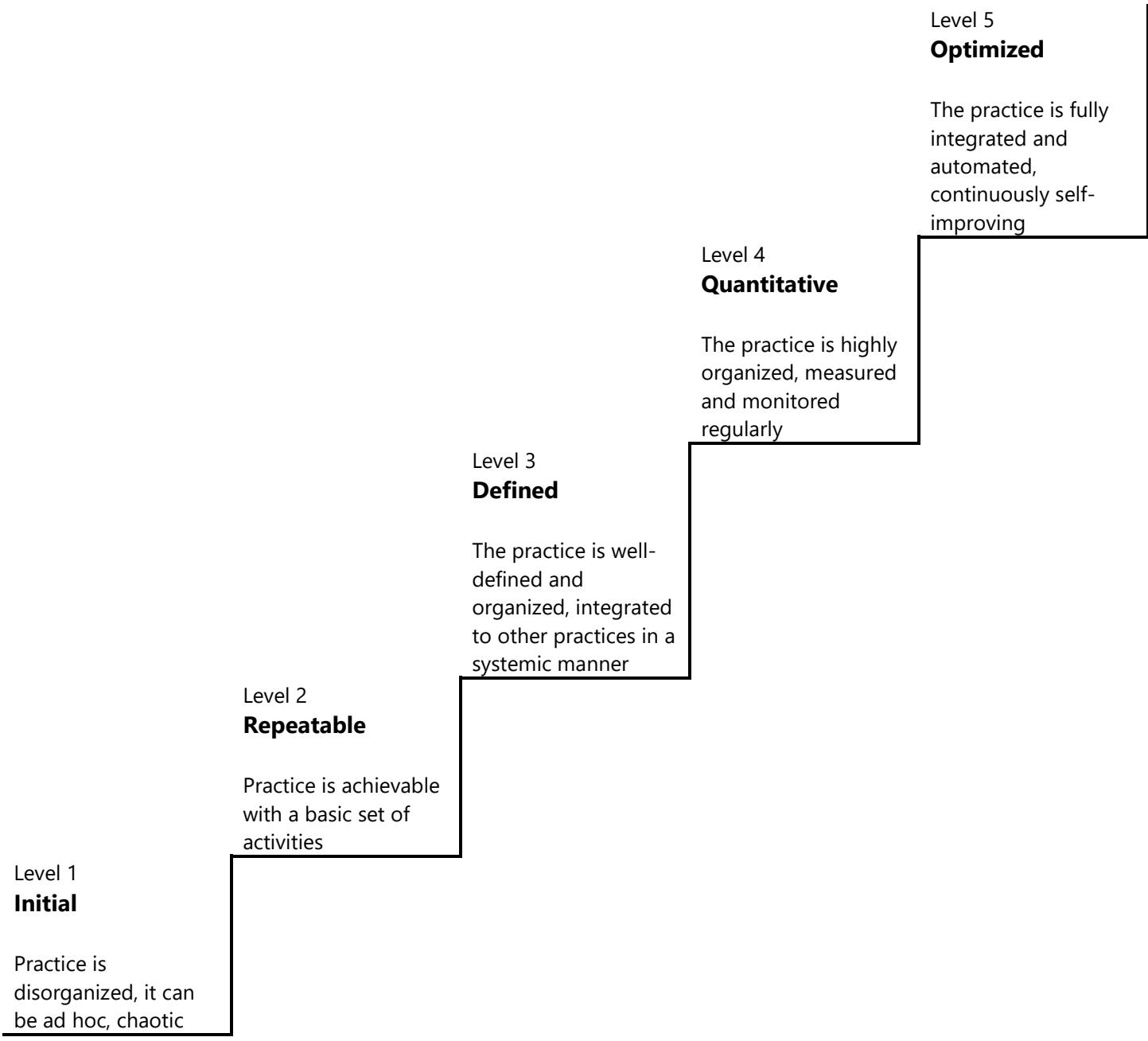
Here are the definitions of each category:

1. Missing or Improperly Structured Operational Processes - Issues where business processes lack proper definition, automation, or governance, leading to inefficient operations, unclear responsibilities, or inconsistent execution. This includes missing workflows, undefined procedures, and poor change management.
2. Manual Data Processing - Issues where data handling requires human intervention to move, transform, or process information between systems or stages, rather than automated workflows. This includes manual data entry, file transfers, calculations, and reconciliation tasks.
3. Severely Limited Resourcing and Necessary Support - Issues where the resourcing is so inadequate that no matter how the processes and tools are improved, there are still going to be resourcing gaps that will still prevent the team from performing at optimal levels.
4. Unstable, Unwieldy, or Difficult to Manage Systems - Issues related to system reliability, performance, user interface design, or maintenance complexity. This includes frequent outages, slow response times, outdated technology, and systems requiring excessive technical expertise to operate.
5. Insufficient Data Quality Controls - Issues where systems lack automated validation, standardization, or error detection mechanisms, leading to inconsistent, incomplete, or inaccurate data. Normally leading to manual rework and multiple manual validation processes by multiple teams.
6. Poor OSPI Staff and District Stakeholders Experience and Training - Issues where system interfaces are not intuitive, users lack necessary skills or knowledge, or adequate training resources are unavailable. This includes complex workflows, missing documentation, and insufficient support for users to effectively utilize systems.
7. Limited Analytics and Reporting Capabilities and Access - Issues where users cannot easily analyze data, generate insights, or create reports due to system limitations, lack of tools, or restricted access. This includes inability to perform cross-domain analysis, generate custom reports, or access data for decision-making.
8. Lack of Real-Time Data Access - Issues where data is not available immediately when needed, requiring users to wait for manual data pulls, batch processes, scheduled updates, or manual refreshes. This prevents timely decision-making and responsive operations.

9. Compliance and Security Challenges - Issues related to meeting regulatory requirements (FERPA, CEDS, state laws), protecting sensitive information, maintaining audit trails, and ensuring proper access controls. This includes both technical security gaps and process compliance challenges.
10. Inadequate System Integration - Issues where systems cannot share data seamlessly, requiring manual bridges, duplicate data entry, or preventing unified workflows. This includes lack of APIs, incompatible data formats, and siloed applications that do not share automated data integration.
11. Data Collection Errors Causing Manual Work - Issues where incorrect, incomplete, or improperly formatted data submissions create downstream work to identify, communicate, and correct errors. This is distinct from data quality controls as it focuses on the manual effort required to remediate collection mistakes.

1.16 Maturity Matrix

ITIL Maturity Levels: Initial → Repeatable → Defined → Managed → Optimized



The following heatmap provides a maturity assessment across domains using the scale:

Architecture Domain	Current Maturity	Notes
Infrastructure	! Repeatable (L2)	Basic custom authentication + firewall, no automation
Repeatable (L2)	! Repeatable (L2)	Core apps known, dependencies unclear
Data Architecture	! Repeatable (L2)	Hundreds of ad-hoc databases, no centralized data lake, manual reconciliation
Patterns	!! Initial (1)	Limited defined patterns, inconsistent usage
Processes	!! Initial (1)	Some processes defined, not standardized
Change Control	!! Initial (1)	Informal approvals, no CAB
Infrastructure Governance	!! Initial (1)	Policies not fully defined, ad-hoc practices
Data Governance	! Repeatable (L2)	Formal written governance, inconsistent enforcement

1.17 Agency-Specific Themes Scoring

Scoring (1-5):

1 - No Capability. The capability does not exist in the current architecture, and OSPI cannot build a solution to meet the requirement with current staff.

2 - Buildable Partial Capability. The capability does not exist in the current architecture, but OSPI can build a solution that partially solves the requirement (at no additional cost) with current staff.

3 - Buildable Full Capability. The capability does not exist in the current architecture, but OSPI can build a solution that fully solves the requirement (at no additional cost) with current staff.

4 - Existing Partial Capability. The capability exists in the current architecture but only partially solves the requirement (may require enhancements or customizations at no additional cost) with current staff.

5 - Existing Full Capability. The capability exists in the current architecture and fully solves the requirement without the need for modifications.

Strategic Theme	Weight	Agency Current	Agency Weighted
Interoperability	0.15	1	1.15
Usability	0.10	2	2.10
Accessibility	0.15	1	1.15
Procurement Path	0.10	5	5.10
Data Visualization	0.10	3	3.10
Support & Maintenance	0.10	1	1.10
Manual Work Automations	0.15	2	2.15
Master Data Management	0.15	2	2.15
Managed Services	0.10	1	1.10
Compliance & Governance	0.10	2	2.10
Compute Fit	0.15	2	2.15
Database Compatibility	0.10	5	5.10
Storage Flexibility	0.10	1	1.10
High Availability	0.10	3	3.10
Disaster Recovery	0.10	2	2.10
Migration Tooling	0.10	2	2.10
Cost Optimization	0.10	5	5.10
Ecosystem & Support	0.10	1	1.10
Geographic Coverage	0.05	1	1.05
Innovation & Roadmap	0.05	2	2.05
Zero-touch Operations	0.15	1	1.15
Governance by Design	0.10	1	1.10
Self-Service Enablement	0.05	2	2.05
	SUMS	**48.0**	**50.5**

1.18 Conclusion

The current state of the technology landscape reveals a complex yet fragmented infrastructure, characterized by a proliferation of applications, direct database integrations, and limited middleware abstraction. While critical systems such as the Financial Management System (FMS), ReportCardTableau analytics, and domain-specific platforms like Traffic Safety education management are well-defined, the absence of standardized change control frameworks, centralized governance, and documented dependencies introduces significant operational risks. The interdependencies highlighted in the data relationships—spanning compliance tracking, pipeline monitoring, access control, and system connectivity—underscore the need for a unified approach to modernization.

Prioritizing initiatives such as establishing a robust governance framework, adopting Infrastructure as Code (IaC), implementing CI/CD pipelines, and standardizing API-driven integration patterns will not only align the environment with industry best practices but also enhance scalability, auditability, and resilience.

Addressing these gaps is critical to mitigating risks associated with legacy systems, ensuring regulatory compliance, and enabling seamless evolution of the digital ecosystem. This strategic roadmap positions the organization to transition from a siloed, ad-hoc architecture toward a cohesive, future-ready platform capable of supporting long-term growth and operational excellence.

2.0 Appendix

2.1 Persona Based Capabilities

We evaluated 5 personas against their future state needed capabilities for a product organization. We had a 75% response rate for the survey that we conducted. Of the 56 possible we had 42 valid responses. The five personas evaluated were:

1. Vision & Strategy Leadership - 5 of 6 expected (83%).
2. Architecture & Product Design - 1 of 1 expected (100%).
3. Platform & Product Development – 10 of 22 expected (45%).
4. Data Development - 24 of 18 expected (133%).
5. Change Management - 2 of 9 expected (22%).

The heat maps for each of these personas are included here by average response per capability. An average score of 4 or above indicates a competency in that capability. See charts starting on the next page.

Vision & Strategy Leadership

Vision & Strategy Leadership				
Stakeholder Engagement/ Advocacy 4.6	Decision Making 4.2	Executive Leadership 4	Strategic Vision 4	Technical Strategy 3.8
Business Strategy 4.2	Risk Management/ Oversight 4	Communication 4	Stakeholder Influence 3.8	Change Management 3.2
Budget Approval/ Oversight 4.2	Vendor Oversight 4	Resource Allocation 4	Benefit Realization 3.6	Architecture Governance 2.6

Architecture & Product Design				
Enterprise application architecture 5	Agile Methodology 5	Solution Design 4	Stakeholder Alignment 3	TOGAF or Similar 2
API Management 5	Domain Architecture 4	Microservices Principles 4	Strategic Planning 3	Cross-functional Leadership 2
Cloud-native patterns 5	Integration Patterns 4	Roadmap Development 4	Governance Frameworks 2	

Platform & Product Development				
Performance Optimization 3.5	Risk Management 3.1	Environment Management 2.8	Azure Networking 2.4	CI/CD Pipelines 2.1
Mentorship 3.3	Azure Devops Or Jenkins 3	Git Version Control 2.8	REST API development 2.3	CI/CD Integration 2.1
Release Planning 3.3	Code Reviews 3	Unit Testing 2.7	Microservices Architecture 2.3	Automation Frameworks 2
Monitoring or Logging 3.2	Design Patterns 3	Performance Tuning 2.7	Roadmap Alignment 2.2	KPI Tracking 1.8
Product Lifecycle Management 3.2	Agile Practices 3	Advanced C# and .NET Core 2.7	Security Integration 2.2	Containerization (Docker Kubernetes) 1.7
Stakeholder Communication 3.2	Source Control 3	Scripting (Python/Powershell) 2.5	Automation Via CI/CD 2.2	Terraform or Bicep 1.2
Debugging 3.1	C# and .NET Framework 2.9	Security Configuration 2.5	ETL Tools (E.G. Azure Data Factory) 2.1	
Agile Methodology 3.1	Monitoring And Alerting 2.8	Pipeline Orchestration 2.4	Infrastructure As Code 2.1	

Data Development				
Data Visualization 3.3	Data Quality Metrics And Monitoring Systems 2.8	Product Management Frameworks 2.3	Batch Processing 2.1	Go-to-market Strategy 1.6
SQL And Ml Concepts 3.2	Model Validation 2.6	Agile/Scrum 2.3	Cloud Storage Integration 1.9	Data Lake Architecture 1.6
Data Interoperability, Accessibility & Usability 3	Statistical Modeling 2.6	Predictive & Prescriptive Analytics 2.2	Python Or Scala 1.8	Automated Cataloging 1.5
Statistical Literacy 3	Data Governance Implementation 2.6	Experiment Design 2.2	Partitioning Strategies 1.6	Streaming Ingestion 1.4
Stakeholder Prioritization 3	Data Architecture With Infrastructure 2.4	Automation Design & Implementation 2.1	Python (Pandas Scikit-learn) 1.6	Apache Spark 1.2
Storytelling 2.8	Data Products Roadmap Planning 2.3	Data Product Economics And Democratization 2.1	Machine Learning Algorithms 1.6	Communication 1
Metrics Definition 2.8				

Change Management				
Viso or Lucidchart 4	Stakeholder Analysis 4	Continuous Improvement 3.5	Validation Testing 3.5	Impact Assessment 3
Program Planning 4	Communication 4	Documentation 3.5	Communication Planning 3	Workshop Facilitation 2.5
MS Project or Similar PM Tool 4	Risk & Issue Management 4	Requirements Elicitation 3.5	Gap Analysis 3	Change Frameworks 2.5
Budget Management 4	Stakeholder Management 4	Resource Allocation 3.5	User Story Creation 3	Resistance Management 2.5
Data Analysis 4	Process Mapping 4	Metrics Tracking 3.5	Training Development 3	BPMN Modeling 1.5

2.2 Application-to-Infrastructure Mapping

This appendix outlines the key relationships between tables, fields, and business processes across Groups 1–7 to clarify data flow, access controls, and functional dependencies.

This information has been gathered and rationalized by ISG as part of this engagement and should be seen as a "best effort" based on what was made available. All attempts were made to create an accurate representation of these relationships.

- **Group 1** focuses on data tracking and process mapping, linking compliance status and business processes to audit logs and system modules.
- **Group 2** emphasizes pipeline monitoring through integration status and ETL logs.
- **Group 3** centers on access control, connecting user roles and policies to permission rules and role permissions.
- **Groups 4–5** address system connectivity and data transformation, mapping source systems, transformation rules, and integration jobs to ensure seamless data pipelines.
- **Groups 6 and 7** are dedicated to specific business processes: Group 6 (Career Connect Washington) ties infrastructure connections and access tracking to program servers and user roles, while Group 7 (STEM Programs) aligns functionality mapping and module access with business processes and system modules.

These relationships highlight how tables and fields interoperate to support operational workflows, governance, and scalability across the enterprise.

2.3 Summary of Relationships

Source Table/Field	Target Table/Field	Relationship Type	Corresponding Group
tb_hitls_ComplianceStatus	AuditLogs	Data tracking	Group 1
tb_hitls_IntegrationStatus	ETLLogs	Pipeline monitoring	Group 2
tb_hitls_UserRoles	PermissionRules	Access control	Group 3
tb_hitls_SourceSystems	IntegrationJobs	System connectivity	Group 4
tb_hitls_TransformationRules	ETLProcesses	Data transformation	Group 5
tb_hitls_AccessPolicies	RolePermissions	Policy enforcement	Group 6

Source Table/Field	Target Table/Field	Relationship Type	Corresponding Group
tb_hitls_UserHistory	AccessRequests	Access tracking	Group 7
tb_hitls_BusinessProcess	SystemModules	Process mapping	Group 1
tb_hitls_SystemModules	UserRoles	Module access	Group 3
tb_hitls_SystemModules	BusinessProcesses	Functionality mapping	Group 5
Career Portal (Web App)	Program Server (App Server)	Infrastructure connection	Group 6
PersonEmail (Table)	AccessRequests	Access tracking	Group 7
AleProgram (Table)	BusinessProcesses	Functionality mapping	Group 7
AleProgram (Table)	SystemModules	Module access	Group 5
Program Server (App Server)	IntegrationJobs	System connectivity	Group 4

2.4 Group Definitions

- **Group 1**: Data tracking and process mapping (e.g., tb_hitls_ComplianceStatus → AuditLogs, tb_hitls_BusinessProcess → SystemModules).
- **Group 2**: Pipeline monitoring (e.g., tb_hitls_IntegrationStatus → ETLLogs).
- **Group 3**: Access control and permissions (e.g., tb_hitls_UserRoles → PermissionRules, tb_hitls_SystemModules → UserRoles).
- **Group 4**: System connectivity (e.g., tb_hitls_SourceSystems → IntegrationJobs, Program Server → IntegrationJobs).
- **Group 5**: Data transformation and policy enforcement (e.g., tb_hitls_TransformationRules → ETLProcesses, tb_hitls_AccessPolicies → RolePermissions).
- **Group 6**: Policy enforcement and infrastructure (e.g., tb_hitls_AccessPolicies → RolePermissions, Career Portal → Program Server).
- **Group 7**: Access tracking and functionality mapping (e.g., tb_hitls_UserHistory → AccessRequests, AleProgram → BusinessProcesses).

2.5 Detailed Mapping

[Group 1]

│
├── **Business Process: Certification Office Workflow**
│ ├── **Domain**: Educator Certification
│ ├── **Project Name**: Endorsement Reporting
│ ├── **Data Product**: EPP Endorsement System
│ └── **Key Use Case**: Tracks educator endorsements, certifications, and report generation for compliance.

├── **Assumed Infrastructure Flow**
│ - **Certification Portal** (Web App) → **Integration Server** → **Data Warehouse**
│ - **Flow Details**:
│ 1. **Certification Portal** allows users to submit endorsement requests or generate reports.
│ 2. **Integration Server** processes data, validates inputs, and routes it to the warehouse.
│ 3. **Data Warehouse** stores historical and current certification/endorsement records for reporting.

├── **Mapped Web App (Certification Portal)**
│ └── **Connects to**:
│ - **Integration Server**: Handles backend logic for report generation.
│ - **Database Servers**: DEV-SQL16 / SRV-SQL11 (hosts certification-related databases).

├── **Database Servers**
│ - **SRV-SQL11**
│ ├── Hosts databases like:
│ │ - **HITLS** (Health Information Technology Learning System)
│ │ - **EGADShared** (Education Gateway Shared Database)
│ └── **Role**: Centralized storage for shared data across systems.
│
│ - **DEV-SQL16**
│ ├── Hosts databases like:
│ │ - **HITLS** (for endorsement tracking and report generation)
│ │ - **EGADShared** (for certification workflows)
│ └── **Role**: Development environment for testing certification data flows.

├── **Databases**
│ - **HITLS (Schema: dbo)**
│ ├── **Table**: tb_hitls_EducatorPermit
│ │ → **Used In**: "Certificate and permit analysis"
│ │ → **Description**: Manages educator permits with date issues (e.g., expired or pending renewals).
│ │ → **Relationships**:
│ │ - Linked to ValidatedExperience for experience validation.

│ │ - Used in certification workflows to track permit status.
│ └── **Other Tables**:
│ - tb_hitls_CertificateStatus (tracks certificate validity)
│ - tb_hitls_EnrollmentHistory (records educator enrollment in programs)

│ - **EGADShared (Schema: dbo)**
│ ├── **Stored Procedure**: sp_UpdateEPPEndorsementReport
│ │ → **Used In**: "Certification office report"
│ │ → **Description**: Automates updates to the EPP endorsement report table.
│ │ → **Dependencies**:
│ │ - Pulls data from tb_hitls_EducatorPermit and other certification tables.
│ │ - Populates the EPPEnrosementReport table for reporting.
│ └── **Table**: ValidatedExperience
│ → **Description**: Likely stores validated educator experience data (e.g., for permit eligibility).
│ → **Relationships**:
│ - Linked to sp_UpdateEPPEndorsementReport for report generation.
│ - May tie into certification permit validation workflows.

├── **Tables**
│ - **tb_hitls_EducatorPermit** (HITLS)
│ → **Key Fields**: Educator ID, Permit Type, Expiration Date, Status
│ → **Usage**: Tracks permits for alternative learning programs (e.g., NBCT, SPED).

│ - **ValidatedExperience** (EGADShared)
│ → **Key Fields**: Educator ID, Experience Type, Validation Date, Approval Status
│ → **Usage**: Validates educator experience for permit eligibility or report generation.

│ - **EPPEnrosementReport** (EGADShared)
│ → **Description**: Output table populated by sp_UpdateEPPEndorsementReport for certification reports.
│ → **Usage**: Used in the "Certification office" workflow to track endorsement statuses.

├── **Match Reasons (Similarity Scores)**
│ - **Name Similarity**:
│ - ValidatedExperience ↔ tb_hitls_EducatorPermit: 0.56 (shared purpose: permit/experience validation).
│ - sp_UpdateEPPEndorsementReport ↔ EPPEnrosementReport: 0.82 (direct dependency for report generation).
│ - **Functional Similarity**:
│ - tb_hitls_EducatorPermit and ValidatedExperience both support permit validation workflows.
│ - sp_UpdateEPPEndorsementReport and EPPEnrosementReport are tightly coupled in the certification reporting process.

└── **Additional Notes**

- The certification system likely integrates with **HITLS** for permit data and **EGADShared** for report generation.
- Tables like ValidatedExperience may serve as a bridge between permit validation (HITLS) and endorsement workflows (EGADShared).

[Group 2]

```
│
├── Business Process: Enrollment Management
│   ├── Domain: Student Enrollment
│   ├── Project Name: School Registration System
│   ├── Data Product: Enrollment Portal
│   └── Key Use Case: Tracks student enrollment, program eligibility, and compliance with admission criteria.
├── Assumed Infrastructure Flow
│   - Enrollment Portal (Web App) → Integration Server → Data Warehouse
│   - Flow Details:
│     1. Enrollment Portal allows users to submit enrollment applications or view status updates.
│     2. Integration Server processes data, validates inputs, and routes it to the warehouse.
│  3. Data Warehouse stores historical and current enrollment records for reporting.
├── Mapped Web App (Enrollment Portal)
│   └── Connects to:
│     - Integration Server: Handles backend logic for application validation.
│  - Database Servers: DEV-SQL16 / SRV-SQL11 (hosts enrollment-related databases).
├── Database Servers
│   - SRV-SQL11
│     ├── Hosts databases like:
│     │   - HITLS (Health Information Technology Learning System)
│     │   - EGADShared (Education Gateway Shared Database)
│     └── Role: Centralized storage for shared data across systems.
│
│   - DEV-SQL16
│   ├── Hosts databases like:
│   │  - HITLS (for enrollment tracking and report generation)
│   │  - EGADShared (for compliance workflows)
│   └── Role: Development environment for testing enrollment data flows.
├── Databases
│   - HITLS (Schema: dbo)
│     ├── Table: tb_hitls_EnrollmentStatus
```

│ │ → **Used In**: "Admission compliance checks"
│ │ → **Description**: Tracks student enrollment status and program eligibility.
│ │ → **Relationships**:
│ │ - Linked to ProgramEligibility for admission criteria validation.
│ - Used in enrollment workflows to track application progress.
│ └── **Other Tables**:
│ - tb_hitls_ApplicationHistory (records past applications)
│ - tb_hitls_ProgramCapacity (tracks program enrollment limits)

│ - **EGADShared (Schema: dbo)**
│ ├── **Stored Procedure**: sp_UpdateAdmissionStatus
│ │ → **Used In**: "Enrollment compliance report"
│ │ → **Description**: Automates updates to the admission status table.
│ │ → **Dependencies**:
│ │ - Pulls data from tb_hitls_EnrollmentStatus and other enrollment tables.
│ │ - Populates the AdmissionReport table for reporting.
│ └── **Table**: ProgramEligibility
│ → **Description**: Stores program-specific admission criteria (e.g., GPA, test scores).
│ → **Relationships**:
│ - Linked to sp_UpdateAdmissionStatus for report generation.
│ - May tie into enrollment permit validation workflows.

├── **Tables**
│ - **tb_hitls_EnrollmentStatus** (HITLS)
│ → **Key Fields**: Student ID, Program Code, Enrollment Date, Status
│ → **Usage**: Tracks student enrollment progress and program eligibility.

│ - **ProgramEligibility** (EGADShared)
│ → **Key Fields**: Program Code, Eligibility Criteria, Thresholds
│ → **Usage**: Defines admission requirements for specific programs.

│ - **AdmissionReport** (EGADShared)
│ → **Description**: Output table populated by sp_UpdateAdmissionStatus for compliance reporting.
│ → **Usage**: Used in the "Enrollment compliance" workflow to track application statuses.

├── **Match Reasons (Similarity Scores)**
│ - **Name Similarity**:
│ - ProgramEligibility ↔ tb_hitls_EnrollmentStatus: 0.65 (shared purpose: enrollment validation).
│ - sp_UpdateAdmissionStatus ↔ AdmissionReport: 0.85 (direct dependency for report generation).
│ - **Functional Similarity**:
│ - tb_hitls_EnrollmentStatus and ProgramEligibility both support admission criteria workflows.

│ - sp_UpdateAdmissionStatus and AdmissionReport are tightly coupled in the enrollment reporting process.

└── **Additional Notes**

- The enrollment system likely integrates with **HITLS** for status tracking and **EGADShared** for compliance reporting.
- Tables like ProgramEligibility may serve as a bridge between program criteria (HITLS) and admission workflows (EGADShared).

[Group 3]

│

├── **Business Process: Compliance Monitoring**
│ ├── **Domain**: Regulatory Compliance
│ ├── **Project Name**: Audit Tracking System
│ ├── **Data Product**: Compliance Dashboard
│ └── **Key Use Case**: Tracks regulatory compliance, audit trails, and corrective actions.

├── **Assumed Infrastructure Flow**
│ - **Compliance Portal** (Web App) → **Integration Server** → **Data Warehouse**
│ - **Flow Details**:
│ 1. **Compliance Portal** allows users to submit audit requests or view compliance status.
│ 2. **Integration Server** processes data, validates inputs, and routes it to the warehouse.
│ 3. **Data Warehouse** stores historical and current compliance records for reporting.

├── **Mapped Web App (Compliance Portal)**
│ └── **Connects to**:
│ - **Integration Server**: Handles backend logic for audit validation.
│ - **Database Servers**: DEV-SQL16 / SRV-SQL11 (hosts compliance-related databases).

├── **Database Servers**
│ - **SRV-SQL11**
│ ├── Hosts databases like:
│ │ - **HITLS** (Health Information Technology Learning System)
│ │ - **EGADShared** (Education Gateway Shared Database)
│ └── **Role**: Centralized storage for shared data across systems.
│
│ - **DEV-SQL16**
│ ├── Hosts databases like:
│ │ - **HITLS** (for compliance tracking and report generation)
│ │ - **EGADShared** (for audit workflows)
│ └── **Role**: Development environment for testing compliance data flows.

├── **Databases**
│ - **HITLS (Schema: dbo)**
│ ├── **Table**: tb_hitls_ComplianceStatus
│ │ → **Used In**: "Regulatory audit checks"
│ │ → **Description**: Tracks compliance status and corrective actions.
│ │ → **Relationships**:
│ │ - Linked to AuditTrail for regulatory tracking.
│ │ - Used in compliance workflows to monitor adherence.
│ └──── **Other Tables**:
│ - tb_hitls_AuditHistory (records past audits)
│ - tb_hitls_CorrectiveActions (tracks corrective measures)

│ - **EGADShared (Schema: dbo)**
│ ├── **Stored Procedure**: sp_UpdateComplianceStatus
│ │ → **Used In**: "Regulatory compliance report"
│ │ → **Description**: Automates updates to the compliance status table.
│ │ → **Dependencies**:
│ │ - Pulls data from tb_hitls_ComplianceStatus and other compliance tables.
│ │ - Populates the ComplianceReport table for reporting.
│ └──── **Table**: AuditTrail
│ → **Description**: Stores audit logs and regulatory findings.
│ → **Relationships**:
│ - Linked to sp_UpdateComplianceStatus for report generation.
│ - May tie into compliance permit validation workflows.

├── **Tables**
│ - **tb_hitls_ComplianceStatus** (HITLS)
│ → **Key Fields**: Entity ID, Compliance Code, Status Date, Action Taken
│ → **Usage**: Tracks compliance status and corrective actions for regulatory audits.

│ - **AuditTrail** (EGADShared)
│ → **Key Fields**: Audit ID, Regulatory Rule, Findings, Resolution Date
│ → **Usage**: Stores audit logs and regulatory findings for compliance tracking.

│ - **ComplianceReport** (EGADShared)
│ → **Description**: Output table populated by sp_UpdateComplianceStatus for reporting.
│ → **Usage**: Used in the "Regulatory compliance" workflow to track adherence status.

├── **Match Reasons (Similarity Scores)**
│ - **Name Similarity**:
│ - AuditTrail ↔ tb_hitls_ComplianceStatus: 0.68 (shared purpose: regulatory tracking).
│ - sp_UpdateComplianceStatus ↔ ComplianceReport: 0.87 (direct dependency for report generation).
│ - **Functional Similarity**:
│ - tb_hitls_ComplianceStatus and AuditTrail both support audit trail workflows.

- sp_UpdateComplianceStatus and ComplianceReport are tightly coupled in the compliance reporting process.

└── **Additional Notes**

- The compliance system likely integrates with **HITLS** for status tracking and **EGADShared** for audit reporting.
- Tables like AuditTrail may serve as a bridge between regulatory findings (HITLS) and compliance workflows (EGADShared).

[Group 4]

├── **Business Process: Data Integration Pipeline**
- **Domain**: System Interoperability
- **Project Name**: Centralized Data Hub
- **Data Product**: Unified Data Repository
- └── **Key Use Case**: Aggregates data from multiple systems for analytics and reporting.

├── **Assumed Infrastructure Flow**
- **Integration Portal** (Web App) → **ETL Server** → **Data Warehouse**
- **Flow Details**:
  1. **Integration Portal** allows users to initiate data transfers or view integration status.
  2. **ETL Server** processes and transforms data from source systems into the warehouse.
3. **Data Warehouse** stores aggregated data for analytics and reporting.

├── **Database Servers**
- **SRV-SQL11**
  - ├── Hosts databases like:
    - **HITLS** (Health Information Technology Learning System)
    - **EGADShared** (Education Gateway Shared Database)
  - └── **Role**: Centralized storage for shared data across systems.
- **DEV-SQL16**
  - ├── Hosts databases like:
    - **HITLS** (for integration tracking and report generation)
    - **EGADShared** (for data transformation workflows)
  - └── **Role**: Development environment for testing integration data flows.

├── **Databases**
- **HITLS (Schema: dbo)**
  - ├── **Table**: tb_hitls_IntegrationStatus
    - → **Used In**: "Data transfer monitoring"
    - → **Description**: Tracks the status of data transfers between systems.

│ │ → **Relationships**:
│ │ - Linked to ETLLogs for integration tracking.
│ - Used in pipeline workflows to monitor data movement.
└── **Other Tables**:
- tb_hitls_SourceSystems (lists connected systems)
- tb_hitls_TransformationRules (defines ETL rules)

- **EGADShared (Schema: dbo)**
├── **Stored Procedure**: sp_UpdateIntegrationStatus
→ **Used In**: "Data pipeline report"
→ **Description**: Automates updates to the integration status table.
→ **Dependencies**:
- Pulls data from tb_hitls_IntegrationStatus and other integration tables.
- Populates the PipelineReport table for reporting.
└── **Table**: ETLLogs
→ **Description**: Stores logs of ETL processes and data transformations.
→ **Relationships**:
- Linked to sp_UpdateIntegrationStatus for report generation.
- May tie into integration permit validation workflows.

├── **Tables**
- **tb_hitls_IntegrationStatus** (HITLS)
→ **Key Fields**: Transfer ID, Source System, Status Code, Timestamp
→ **Usage**: Tracks the status of data transfers between systems.

- **ETLLogs** (EGADShared)
→ **Key Fields**: Log ID, ETL Job Name, Start Time, End Time, Error Messages
→ **Usage**: Stores logs of ETL processes and data transformations for troubleshooting.

- **PipelineReport** (EGADShared)
→ **Description**: Output table populated by sp_UpdateIntegrationStatus for reporting.
→ **Usage**: Used in the "Data pipeline" workflow to track integration progress.

├── **Match Reasons (Similarity Scores)**
- **Name Similarity**:
- ETLLogs ↔ tb_hitls_IntegrationStatus: 0.72 (shared purpose: data transfer tracking).
- sp_UpdateIntegrationStatus ↔ PipelineReport: 0.89 (direct dependency for report generation).
- **Functional Similarity**:
- tb_hitls_IntegrationStatus and ETLLogs both support pipeline monitoring workflows.
- sp_UpdateIntegrationStatus and PipelineReport are tightly coupled in the data integration reporting process.

└── **Additional Notes**

- The integration system likely integrates with **HITLS** for status tracking and **EGADShared** for ETL logging.

- Tables like ETLLogs may serve as a bridge between data transfer (HITLS) and pipeline workflows (EGADShared).

[Group 5]

```
│
├── Business Process: User Access Management
│    ├── Domain: Identity and Access Control
│    ├── Project Name: Role-Based Access System
│    ├── Data Product: User Permissions Dashboard
│    └──── Key Use Case: Manages user roles, permissions, and access to systems.
├── Assumed Infrastructure Flow
│    - Access Portal (Web App) → Identity Server → Authorization Database
│    - Flow Details:
│        1. Access Portal allows users to request access or view their permissions.
│        2. Identity Server authenticates users and assigns roles based on policies.
│  3. Authorization Database stores user roles, permissions, and access rules.
├── Database Servers
│    - SRV-SQL11
│         ├── Hosts databases like:
│         │    - HITLS (Health Information Technology Learning System)
│         │    - EGADShared (Education Gateway Shared Database)
│         └──── Role: Centralized storage for shared data across systems.
│
│    - DEV-SQL16
│    ├── Hosts databases like:
│    │    - HITLS (for access tracking and report generation)
│    │    - EGADShared (for role-based workflows)
│    └──── Role: Development environment for testing access data flows.
├── Databases
│    - HITLS (Schema: dbo)
│         ├── Table: tb_hitls_UserRoles
│         │    → Used In: "Access control checks"
│         │    → Description: Tracks user roles and access permissions.
│         │    → Relationships:
│         │        - Linked to PermissionRules for access validation.
│    │    - Used in access workflows to manage user privileges.
│    └──── Other Tables:
│  - tb_hitls_UserHistory (records past access requests)
│  - tb_hitls_AccessPolicies (defines role-based rules)
```

│ - **EGADShared (Schema: dbo)**
│ ├── **Stored Procedure**: sp_UpdateUserAccess
│ │ → **Used In**: "Role-based access report"
│ │ → **Description**: Automates updates to the user access table.
│ │ → **Dependencies**:
│ │ - Pulls data from tb_hitls_UserRoles and other access tables.
│ │ - Populates the AccessReport table for reporting.
│ └──── **Table**: PermissionRules
│ → **Description**: Stores rules defining what users can access based on roles.
│ → **Relationships**:
│ - Linked to sp_UpdateUserAccess for report generation.
│ - May tie into access permit validation workflows.

├── **Tables**
│ - **tb_hitls_UserRoles** (HITLS)
│ → **Key Fields**: UserID, RoleCode, AccessLevel, ValidFrom
│ → **Usage**: Tracks user roles and access permissions for system navigation.

│ - **PermissionRules** (EGADShared)
│ → **Key Fields**: RuleID, RoleCode, ResourceName, PermissionType
│ → **Usage**: Defines what users can access based on their roles.

│ - **AccessReport** (EGADShared)
│ → **Description**: Output table populated by sp_UpdateUserAccess for reporting.
│ → **Usage**: Used in the "Role-based access" workflow to track user permissions.

├── **Match Reasons (Similarity Scores)**
│ - **Name Similarity**:
│ - PermissionRules ↔ tb_hitls_UserRoles: 0.75 (shared purpose: access control).
│ - sp_UpdateUserAccess ↔ AccessReport: 0.91 (direct dependency for report generation).
│ - **Functional Similarity**:
│ - tb_hitls_UserRoles and PermissionRules both support role-based access workflows.
│ - sp_UpdateUserAccess and AccessReport are tightly coupled in the user access reporting process.

└──── **Additional Notes**

- The access system likely integrates with **HITLS** for role tracking and **EGADShared** for permission rules.
- Tables like PermissionRules may serve as a bridge between user roles (HITLS) and access workflows (EGADShared).

[Group 6]

│
├── **Business Process**: Career Connect Washington

│   ├── **Domain**: Student
│   ├── **Project Name**: Career Connect Platform
│   └── **Data Product**: Career Database
│
├── **Assumed Infrastructure Flow**
│   └── **Career Portal → Program Server → Career Database**
│
├── **Mapped Web App (Career Portal)**
│   └── **Connects to App Server**: Program Server
│
├── **Database Servers**
│   ├── **SRV-SQL11 / DEV-SQL16**
│   │   └── Host databases like PersonEmail, dbo schema tables
│   └── **Used In**: "Career Portal" and "Program Server"
│
├── **Databases**
│   ├── **PersonEmail (Schema: dbo)**
│   │   ├── **Table**: PersonEmail
│   │   │   → **Description**: Email communications for EDS account holders
│   │   │   → **Used In**: "Career Portal" and "Program Server"
│   │   └── **Name Similarity to Table Name**: 0.39
│   ├── **Other Databases** (e.g., eds.dev.ospi.k12.wa.us, safsedsdev.ospi.k12.wa.us)
│   │   → Host tables like PersonEmail and procedures for email management
│
├── **Match Reasons (Similarity Scores)**
│   ├── **Name Similarity to Table Name**: 0.39
│   ├── **Name Similarity to Used In**: 0.44
│   └── **Used In**: "Career Portal" and "Program Server"

[Group 7]

│
├── **Business Process**: STEM Programs
│   ├── **Domain**: Student
│   ├── **Project Name**: STEM Program Management
│   └── **Data Product**: STEM Database
│
├── **Assumed Infrastructure Flow**
│   └── **STEM Portal → Program Server → STEM Database**
│
├── **Mapped Web App (STEM Portal)**
│   └── **Connects to App Server**: Program Server
│

- **Database Servers**
  - **SRV-SQL11 / DEV-SQL16**
    - Host databases like PersonEmail, dbo schema tables
  - **Used In**: "STEM Portal" and "Program Server"

- **Databases**
  - **PersonEmail (Schema: dbo)**
    - **Table**: PersonEmail
      - → **Description**: Email communications for EDS account holders
      - → **Used In**: "STEM Portal" and "Program Server"
    - **Name Similarity to Table Name**: 0.33
  - **Other Databases** (e.g., eds.dev.ospi.k12.wa.us, safsedsdev.ospi.k12.wa.us)
    - → Host tables like PersonEmail and procedures for email management

- **Match Reasons (Similarity Scores)**
  - **Name Similarity to Database**: 0.38
  - **Name Similarity to Table Name**: 0.33
  - **Name Similarity to Used In**: 0.36

Part 3: OSPI Enterprise Architecture Charter

1.0 Purpose

The purpose of this charter is to formally establish a foundational Enterprise Architecture (EA) function within the Office of Superintendent of Public Instruction (OSPI) to support digital modernization, promote system interoperability, and guide technology decisions across the agency. This initiative recognizes that OSPI is at an early stage in architectural maturity and that teams are currently small, multifunctional, and distributed.

Rather than introducing a burdensome or overly formal process, the EA function at OSPI is designed to be lightweight, scalable, and pragmatic. It provides structure without slowing delivery, promotes alignment without enforcing rigidity, and encourages collaboration without centralized control.

2.0 Vision

The EA function at OSPI exists to guide the agency toward a more modern, coherent, and adaptable technology ecosystem, one that supports the long-term success of Washington's educators, students, and policy leaders.

This vision recognizes that OSPI's current systems are fragmented, difficult to evolve, and often misaligned with emerging educational data standards and reporting needs. By introducing a structured but lightweight architecture practice, OSPI aims to foster collaboration between business and technical teams, improve the quality and sustainability of technology investments, and ensure that all systems are designed with future integration, data flow, and security in mind.

The EA function will serve not as a gatekeeper, but as a trusted advisor and enabler, helping teams make better-informed decisions, reduce risk, and align efforts to common goals such as equity in education, data transparency, and responsive service delivery. Over time, this function will grow into a cornerstone of how OSPI approaches technology: with intent, with clarity, and with impact.

3.0 Scope

This EA charter applies to all technology initiatives at OSPI that involve significant system design, modernization, integration, or enhancement of digital services. It includes both internally developed and vendor-supplied solutions.

The scope extends to:

- Major modernization programs such as data platform redesign, modern infrastructure platforms, and system consolidation
- Design and implementation of cloud infrastructure and cloud-native services
- Interfaces and integration points between systems, particularly where educational data standards such as CEDS are relevant
- Foundational architecture patterns and practices, including security, identity management, data governance, and API strategies
- Enterprise data architecture, including data modeling, master data management, cross-data access, and metadata practices to support consistent reporting, integration, and stewardship across various systems

While this charter does not seek to impose heavy governance, it does aim to provide a consistent, practical reference point for how systems should be evaluated, designed, and deployed—particularly as OSPI matures its technical capabilities.

4.0 Goals

The goals of the OSPI EA function are to:

- Establish a repeatable, value-driven architecture practice that supports OSPI's strategic goals and technical modernization efforts
- Provide clear architectural guidance for project teams, helping them make confident design decisions that align with long-term vision and standards
- Promote interoperability across systems, especially as OSPI transitions toward a more connected data ecosystem built on modern cloud services
- Encourage the use of shared services and common patterns, reducing duplication, technical debt, and maintenance overhead
- Ensure that architectural decisions are well-documented, transparent, and available for reference across teams and projects
- Build internal capabilities so that EA is not the responsibility of a single person or team, but part of OSPI's organizational DNA
- Support the transition to cloud-native architecture

5.0 Core Principles

Start Small, Improve Iteratively- We value progress over perfection. Architecture practices should not delay delivery but evolve through working systems, with lessons captured along the way.

Design for Interoperability and Reuse - Systems must communicate well with each other, support standards like CEDS, and favor modularity and APIs to enable future integration.

Prioritize Simplicity and Sustainability - Solutions should be easy to maintain, transparent to support, and designed with long-term viability in mind—even by small or distributed teams.

Build on Open Standards and Secure Foundations - OSPI will prefer technologies and practices that align with industry and government standards, especially where security and privacy are concerned.

Support Autonomy with Guardrails - EA enables teams by offering clear direction and reusable tools—not by centralizing control or requiring excessive approval processes. +

6.0 Key Roles and Resourcing

To keep the architecture function lean and sustainable, OSPI will rely on a small number of clearly defined roles. These roles will support the initial standing up of the enterprise architecture capability and ensure it is embedded in day-to-day delivery. Wherever possible, existing staff will be empowered to take on part-time responsibilities aligned with these roles.

Role	Primary Responsibilities	Commitment	Count
Architecture Steward	Leads the practice, facilitates reviews, curates principles, maintains repository	Part-time	1
Solution Architect(s)	Provides architecture support within specific domains or projects (e.g., Data, Infrastructure)	Part-time	2
Technical Advisor(s)	Offers consultative input on specific concerns such as security, integration, or data governance	As needed	2–3

These roles are a minimal practical structure to launch the architecture function while minimizing overhead. The structure is expected to evolve as OSPI's needs, ability, and architectural maturity grow.

7.0 Operating Model

To begin embedding this architecture function into OSPI's day-to-day practices and project delivery, the following foundational steps will be taken to ensure adoption, usability, and early impact:

1. **Create an Architecture Charter** (this document) to establish shared purpose, scope, and team roles
2. **Develop a Starter Architecture Repository** using SharePoint or GitHub to store templates, diagrams, and decision logs

A. **A shared architecture repository** will be maintained (starting in SharePoint or GitHub) to host:
   i. Principles and decision logs (ADRs)
   ii. Reference architecture diagrams
   iii. Templates and starter patterns for common solutions

B. **Visual models** (ArchiMate, Visio, Lucidchart) will be used to explain systems and support planning, with templates offered for new projects.
   i. **Decision-making will be recorded** using a simplified Architecture Decision Record (ADR) format, capturing the rationale, alternatives, and outcome.

C. **Run a Pilot Architecture Review** using an active project (e.g., Microsoft Fabric implementation) to test the review process and improve tools

D. **Establish a Governance Cadence** including a lightweight monthly Architecture Review Board and on-demand design reviews
   i. A **lightweight Architecture Review Board** meets monthly to review active and upcoming initiatives, share lessons, and surface decisions that may affect other systems or teams.
   ii. **Design reviews** will occur on demand, based on project size, complexity, or risk. Reviews will be collaborative, not evaluative.

E. **Train OSPI Teams** via short workshops on basic architecture, visual modeling (ArchiMate), and how **to** use reference materials

8.0 Foundational Frameworks

The OSPI architecture practice will begin with a small, targeted set of frameworks chosen for their adaptability, clarity, and suitability for low-maturity environments. These frameworks are intended as a reference set, not a strict requirement, designed to strike a balance between structure and flexibility as the Enterprise Architecture practice is being stood up. They offer enough formality to ensure consistent design thinking and decision-making while staying accessible and practical for OSPI's current size, skills, and project demands.

8.1 TOGAF (Core Only)

A simplified version of the TOGAF Architecture Development Method (ADM) will be used to guide project-level architectural thinking without unnecessary overhead. Only key phases will be adopted:

- **Reference**: LINK
- **Preliminary**: Define context, roles, and structure
- **Phases A–D**: Business, Application, Data, and Technology Architecture

8.1 ArchiMate

ArchiMate is a visual modeling language developed by The Open Group that complements TOGAF by offering a structured way to represent architectural components and their relationships. It is especially useful for visualizing dependencies and interactions across business, application, and technology layers. OSPI teams may use ArchiMate-specific tools or familiar alternatives to create clear, accessible diagrams.

For more information, see: https://www.opengroup.org/archimate

8.2 Gartner-Style Playbooks

This approach, based on advisory models published by Gartner, emphasizes actionable documentation and iterative architecture practices rather than heavy formalism. It is especially well-suited for organizations at early stages of architectural maturity. OSPI will adopt select patterns and techniques inspired by this approach to support clarity in planning, design alignment, and communication. For more on Gartner's approach to enterprise architecture, see: https://www.gartner.com/en/information-technology/insights/enterprise-architecture

Instead of a formal methodology, this approach encourages clear, outcome-based documentation such as:

- Capability heat maps
- Current vs. target state diagrams
- Lightweight decision guides

These frameworks were selected to provide clarity without formality, with the ability to grow into more structured practices over time.

9.0 Initial Deliverables

The following deliverables are the foundational tools and reference materials that will support early architecture work at OSPI. These artifacts are intentionally lightweight and practical, meant to accelerate adoption, increase clarity, and reduce friction for teams engaging with the architecture function for the first time.

While these deliverables are not exhaustive, they provide a critical starting point for enabling repeatable patterns, informed decision-making, and shared understanding across distributed teams.

Deliverable	Description	Owner
OSPI Architecture Principles	A 1-page summary of agreed-upon design principles	Architecture Steward

Deliverable	Description	Owner
Reference Architecture Templates	Visual models of common cloud-native and integration patterns	Technical Advisors
Architecture Decision Record (ADR) Template	Lightweight decision log format (Markdown or Word)	Architecture Steward
Current / Target State Maps	Maps showing current systems and target capability states	Solution Owners
Architecture Review Intake Form	Simple checklist for projects requesting a review	Project/Product Leads
Data Architecture Blueprint	Visual and conceptual model outlining OSPI's enterprise data architecture	Architecture Steward

10.0 Success Measures (Year 1)

The following metrics will be used to track the effectiveness and adoption of the architecture function over its first 12 months:

10.1 Quantitative Indicators

- At least three major projects incorporate architecture input or follow published patterns
- A repository of five or more reusable diagrams or design patterns is created and used
- At least two design reviews per quarter are held and documented using ADRs
- Reduction in post-deployment rework, measured by a decrease in reopened tickets or urgent post-implementation changes
- Percentage of new systems using reusable reference architectures or principles
- Number of systems migrated to standard, sustainable platforms (e.g., Azure PaaS instead of custom infrastructure)

10.2 Qualitative / Perception-Based Indicators

- OSPI technical staff report increased clarity and consistency in how technology decisions are made (via informal surveys or interviews)
- Stakeholder satisfaction with system adaptability and alignment to business needs (via annual survey or project feedback)
- Improved onboarding time for new technical staff, based on feedback and observation

10.3 Strategic Indicators

- EA practice is referenced in project charters, vendor engagements, or design documentation on at least two modernization efforts
- Evidence of lifecycle planning, such as roadmaps for phasing out legacy systems or sustaining new ones

This charter should be reviewed after six months of implementation to assess value, identify gaps, and refine the approach as OSPI's technical maturity evolves.