ISWC System Audit

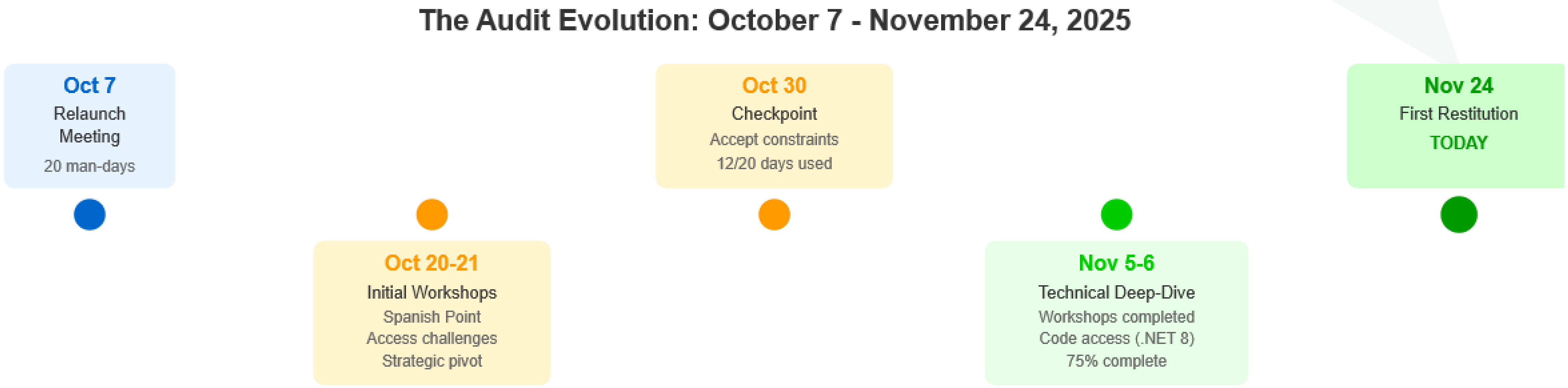
Findings & Strategic Recommendations

Teragone Factory

Bastien Gallay & Guillaume Jay

November 24, 2025

The Audit Journey - How We Got Here



The Audit Evolution: October 7 - November 24, 2025

Oct 7
Relaunch Meeting
20 man-days

Oct 20-21
Initial Workshops
Spanish Point
Access challenges
Strategic pivot

Oct 30
Checkpoint
Accept constraints
12/20 days used

Nov 5-6
Technical Deep-Dive
Workshops completed
Code access (.NET 8)
75% complete

Nov 24
First Restitution
TODAY

Key Statistics

- 20 man-days budget (12 for investigation, 8 synthesis)
- 6+ workshops with Spanish Point
- 343 Azure resources analyzed
- 95+ validation rules documented

Today's agenda

Part 1: The Journey	• How the audit evolved • Timeline and key milestones • The strategic pivot
Part 2: Executive Summary	• Platform technically sound • Governance and control challenges • Three critical findings
Part 3: Technical Findings	• Architecture assessment • Code quality analysis • Technical debt status • Infrastructure & DevOps Performance & scalability • Cost analysis
Part 4: Governance Findings	• Production incident • Before/After CAB implementation • Access challenges timeline • Documentation drift • Transparency issues
Part 5: Vendor Lock-in Reality	• Three lock-in mechanisms • Matching Engine deep-dive • What we can't access • Vendor switch effort estimate

Part 6: Strategic Recommendations	• Recommended hybrid approach • Knowledge transfer pilot test • Contract renegotiation leverage • Technical quick wins • Hyperscale proposal assessment
Part 7: Decision Points & Next Steps	• Five key decisions for CISAC • Immediate next steps • Thank you & discussion
Annex A: Detailed Findings Reference	• Pointers to full documentation
Annex B: Validation Rules Reference	• 95+ validation rules documented
Annex C: Items Needing Verification	• Verified facts vs. assumptions requiring validation

The Strategic Pivot - What This Audit Became

What We Were Asked To Do

✓ Validate Hyperscale proposal

✓ Assess architecture quality

✓ Identify technical debt

✓ Migration roadmap

What We Actually Discovered

⚠️ Cost control gap (€600K/year)

✓ Platform technically sound

✓ Recently upgraded, well-maintained

⚠️ Governance gaps are the real issue

🔴 Vendor independence unknown

🔴 Knowledge transfer HIGH RISK

Key Insight

"The platform is solid. Control is the problem."

PART 2

Executive Summary

Tech and Governance Findings

Positive Findings	Negative Findings
✅ Well designed high level DevOps Architecture ✅ Fully automated CI/CD pipeline	❌ Vendor-Locking Mechanisms ❌ Transparency issues ❌ High costs
✅ Code Well-structured, easy to navigate	❌ Various code smells (duplication, too many imbrications, ..)
✅ Recent Code Upgrades	❌ Net Core 10 release on November 13 ❌ Outdated Front packages, Databricks
	❌ Difficult communications

The Good News - Platform Technically Sound

Category	Assessment
Architecture	✅ Well-designed architecture (12 containers, clean separation) ✅ Matching Engine physically separated (REST API integration) ✅ Azure-native services with industry-standard patterns (Autoscaling, Monitoring alerts)
Code Quality	✅ Well-structured, easy to navigate ✅ Dependency injection, interface abstractions ✅ Clean separation of concerns (Controllers, Services, Repositories)
Recent Upgrades	✅ .NET 3.1 → .NET 8.0 (upgraded Nov 4, 2025) ✅ Entity Framework 3.1 → 8.0
DevOps Maturity	✅ Fully automated CI/CD pipeline (20-30 min build + test + deploy) ✅ 700+ automated tests ✅ Multi-stage deployment (Dev → UAT → Production) ✅ Auto-scaling functioning correctly

The Challenge - Where Control is Missing

🔴 CRITICAL: Vendor Lock-in Mechanisms

- Proprietary components (Matching Engine, SmartAIM)
- High quantity of implicit knowledge

🔴 CRITICAL: Knowledge Transfer Viability Unknown

- Misorganized documentation
- Lack of vendor cooperation with third parties

🔴 CRITICAL: Cost Control Gap

- High level of spending
- Unexplained variations

🟠 HIGH: Governance Gaps

- An history of previous incidents
- Transparency issues with vendor relationship

🟡 MEDIUM: Technical Debt Remaining

- Some components beyond support end of life
- Recurrent code anti-patterns

Three Critical Findings

Finding	Status	Impact	Key Unknown
🔴 **FINDING 1:** Vendor Lock-in is Real, But Manageable	Technically separated, contractually/organizationally coupled	Vendor switch = 12-24 months, €300-600K (preliminary estimate)	Matching Engine alternatives exist?
🔴 **FINDING 2:** Knowledge Transfer Viability is Unknown	Cannot confirm maintainability by third party	Vendor switch may be infeasible regardless of technical quality	Can independent vendor deliver with available materials?
🔴 **FINDING 3:** Cost Control Gap is Solvable	€600K/year with no correlation tooling	Cannot explain spending, cannot forecast, cannot optimize	What are the actual cost drivers month-to-month?

PART 3

Technical Findings

Governance History - Key elements were missing

✅ **Well-structured codebase**	⚠ **Minimal code comments**
Logical project organization	Business logic not documented
Easy to navigate	"Locked in people's heads"

✅ **Clean separation of concerns**	⚠ **Significant code duplication / Imbrication**
Controllers, Services, Repos, Managers	Repeated patterns across modules
Interface abstractions / Dependency Injection	Lots of local functions
Testability enabled	Maintainability Risks

✅ **Modern .NET 8 patterns**	⚠ **High implicit knowledge**
Async/await usage	Contextual understanding required
Entity Framework Core 8.0	No Onboarding document

Technical Debt Status - Mixed Picture

✅ RESOLVED (Recently Addressed)

- .NET Core 3.1 → 8.0 (EOL Dec 2022 → Active support until Nov 2026)
- Entity Framework 3.1 → 8.0 (Modern ORM)
- Outdated packages → Security vulnerabilities addressed

⚠ MEDIUM PRIORITY

- NET 8 support ends Nov 2026 (only 1 year remaining)
- Consider .NET 10 for longer support window
- Code smells
- Documentation drift (specs from 2019-2020)

🔴 URGENT

- Databricks 10.4 LTS outdated (unknown EOL, missing modern features)
- Pipeline test runner blocked post-.NET 8 upgrade (workaround active)
- Test stability issues (some tests brittle, 99.5% pass rate)

🟡 HIDDEN INFORMATION

- Git commit history not provided (compliance review pending)
- IaC templates not included (proprietary Smart AIM library)

Code Smells – A few samples

Code Duplication

For example : Three different Pipeline Managers : only one (non-business) different line

⚠ Synchronous Matching Engine Calls
→ Blocking HTTP calls (performance coupling)
→ Large submissions may cause latency
→ Alternative: Queue-based async pattern would decouple

Code optimization N+1 issues

```
var children = submissions.Where(x => x.MultipleAgencyWorkCodesChild).GroupBy(x => x.SubmissionParentId);

foreach (var child in children)
{
    // N+1 PROBLEM: FirstOrDefault() is called for EACH child
    var parent = submissions?.FirstOrDefault(x => x.SubmissionId == child.FirstOrDefault()?.SubmissionParentId);
    var parentId = child.FirstOrDefault()?.SubmissionParentId;

    // Multiple .Where() queries are executed for each child group
    submissions?.Where(x => x.SubmissionParentId == parentId || ...
    submissions?.Where(x => x.SubmissionParentId == parentId).Select(...)
```

Code Smells – Cosmos DB Efficiency

⚠ Require more investigations (A/B Testing, regression checks, performance monitoring) ⚠

```
// Sequential upserts
await auditContainer.UpsertItemAsync(...);            // Line 70
await auditRequestContainer.UpsertItemAsync(...);     // Line 144
await auditRequestContainer.UpsertItemAsync(...);     // Line 172
// ... many more
```

Problem: Each `UpsertItemAsync` is a separate round-trip to CosmosDB. For 100 submissions with 3 upserts each = 300 separate calls.

Problematic Code Patterns
Found in :

- SearchComponent
- Matching Pipeline
- Metadata Validation
- Static Data Validation
- Post-Matching Validation

…

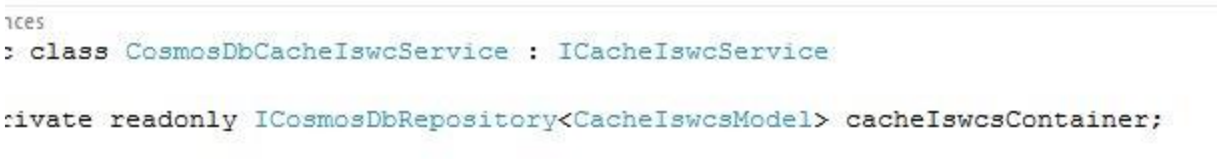
```
foreach (var submission in submissions)
{
    // Line 95-98: Query for MER transactions
    workflowInstanceId = (await
iswcLinkedToRepository.FindManyAsyncOptimizedByPath(...))...

    // Line 103-104: Query for work numbers
    var childIswc = (await
workRepository.FindAsyncOptimizedByPath(...))...

    // Line 107-110: Another workflow query
    workflowInstanceId = (await
iswcLinkedToRepository.FindManyAsyncOptimizedByPath(...))...

    // Line 115-120: Query for DMR
    var iswc = (await
workRepository.FindAsyncOptimizedByPath(...))...
}
```

Problem: Each submission in the batch triggers 2-5 database queries. For 100 submissions, that's 200-500 queries!

Code Smells – Cosmos DB Misusage

⚠ Require more investigations (A/B Testing, regression checks, performance monitoring) ⚠

ISWC Cache implemented in CosmoDb :



```
nces
: class CosmosDbCacheIswcService : ICacheIswcService

:ivate readonly ICosmosDbRepository<CacheIswcsModel> cacheIswcsContainer;
```

⚠ Cost (RU consumption)
⚠ Performance
✅ Standard solution : Redis

Infrastructure / DevOps - Mature Pipeline, IaC Gap

Azure Resource Inventory

343 Resources Across 4 Environments
- Production (primary)
- UAT (user acceptance testing)
- Dev (development)
- Matching Engine (separate environment)

⚠ Not Tagged

App Hosting	App Services
Databases	SQL Server, Cosmos DB
Data Processing	Databricks, Data Factory
Storage	SFTP, Data Lake, Key Vault
Networking	Virtual Network, API Management Gateway ,..
Management	App Insights, Log Analytics
Virtual Machines	38 (Databricks and SFTP

Fully Automated Deployment Pipeline (Build, Test, Deploy)

Infrastructure-as-Code/ DevOps Status: ⚠ **PROPRIETARY**

Spanish Point Position:

- IaC/DevOps templates = "Smart AIM library" (proprietary)
- NOT included in source code delivery
- Licensing program exists for third-party vendors

CISAC Options for Vendor Switch:

1. Full Switch :
   Require Rewriting CI/CD (fairly standard)
2. License Smart AIM library for new vendor (cost unknown)

Performance – Autoscaling everywhere

Historical Performance (Past 12 Months)

✅ No significant issues reported
✅ Auto-scaling handling load variations
⚠️ One exception: ISWCs with 8,000+ works caused SQL exceptions
→ Proposed solution: Pagination for large submissions

Auto-Scaling Configuration

✅ App Services: 2-8 instances (CPU/memory-based)
✅ Cosmos DB: Request Units auto-scale (usage-based)
✅ Databricks: Cluster sizing adjusts to workload

Monitoring & Alerts

✅ Azure Monitor configured (platform-native metrics)
✅ Alerts: CPU >80% for 5 minutes → email to support team
✅ Application Insights available (accessible to audit team)
⚠️ No formal performance SLAs (only incident response time SLAs)
⚠️ No proactive performance dashboards or business KPIs

Analysis - €600K/Year, Visibility Gap

Annual Cloud Spending: ~€600,000 (~€50K/month average)

1. Cosmos DB (Largest portion)
   - Use case: Audit tracking container (append-only logs)
   - Auto-scaling: Request Units vary with usage
   - Concern: Expensive for audit-only use case
2. SQL Server (Business Critical tier)
   - Fixed provisioned capacity
   - Target of Hyperscale proposal (-€1,800/month potential)
3. Databricks (Usage-based)
   - File processing workloads
   - Auto-scaling enabled (cluster runtime costs)
4. Data Factory (Pipeline orchestration)
   - Activity runs, data movement
5. App Services (APIs, portals)
   - Auto-scaling 2-8 instances

Current Process for understanding Cost spikes

- Require MANUAL support ticket investigation
- No proactive monitoring, no automated correlation dashboard

Impact on CISAC

❌ Cannot explain monthly variations to stakeholders
❌ Cannot forecast spending based on expected usage
❌ Cannot identify cost optimization opportunities proactively

PART 4

Governance Findings

Governance History - Key elements were missing

May-June 2024 Production Incident

"Six months of hell, incessant incidents"

What Happened

POC code from unrelated project accidentally merged to production branch and deployed to production

Root Cause

- No branch protection preventing POC merges
- No pre-production testing gate caught the issue
- No rollback executed (or couldn't rollback)
- Governance gap allowed uncontrolled deployment

Impact

- 6 months continuous incident recovery
- User complaints, system instability
- CISAC resources consumed troubleshooting
- Trust in vendor relationship damaged

Recovery

- CAB (Change Advisory Board) established May 2024
- Deployment control and notification process
- Expert group review before production releases

Five years without a CAB !

❌ No deployment tracking
❌ No change notification
❌ No governance oversight
❌ Uncontrolled production releases
❌ "They did everything when they wanted, without warning"

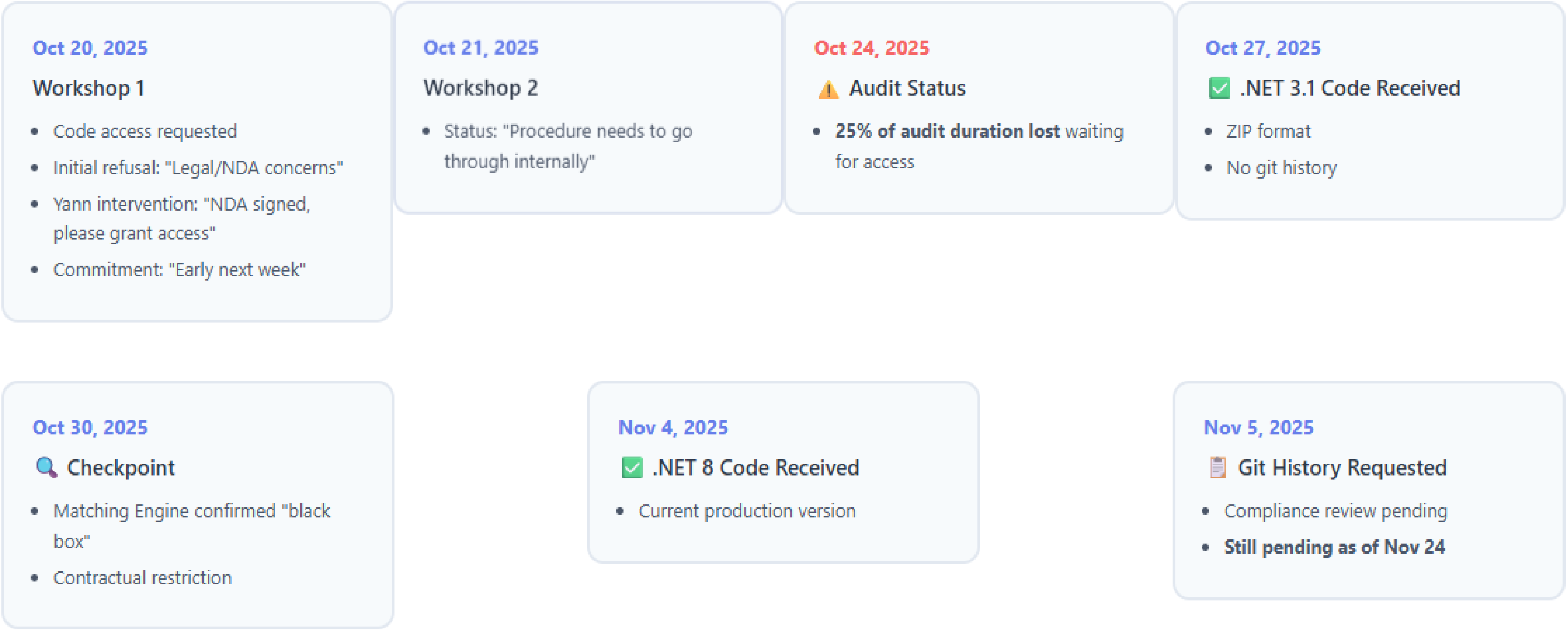
Since May 2024

✅ Deployment history tracking
✅ Controlled deployment schedule
✅ Expert group review
✅ CAB approval required
✅ Deployment control process

Remaining Gaps

⚠ Definition of Done incomplete (documentation updates not required)
⚠ No Architecture Decision Records (ADR)
⚠ Knowledge management processes undefined
⚠ Onboarding processes non-existent

Audit Access Challenge - 25% Audit Duration Lost



Oct 20, 2025

Workshop 1

- Code access requested
- Initial refusal: "Legal/NDA concerns"
- Yann intervention: "NDA signed, please grant access"
- Commitment: "Early next week"

Oct 21, 2025

Workshop 2

- Status: "Procedure needs to go through internally"

Oct 24, 2025

⚠️ Audit Status

- **25% of audit duration lost** waiting for access

Oct 27, 2025

✅ .NET 3.1 Code Received

- ZIP format
- No git history

Oct 30, 2025

🔍 Checkpoint

- Matching Engine confirmed "black box"
- Contractual restriction

Nov 4, 2025

✅ .NET 8 Code Received

- Current production version

Nov 5, 2025

📋 Git History Requested

- Compliance review pending
- **Still pending as of Nov 24**

Access Outcome:

✅ GRANTED: ISWC application source code (.NET 3.1 and .NET8)
🔴 BLOCKED: Matching Engine source code (only on contract termination)
🟡 PENDING: Git commit history (compliance review ongoing)
🔴 EXCLUDED: IaC and CI/CD pipeline definitions (proprietarySmart AIM)

Documentation Drift – Specs from 2019

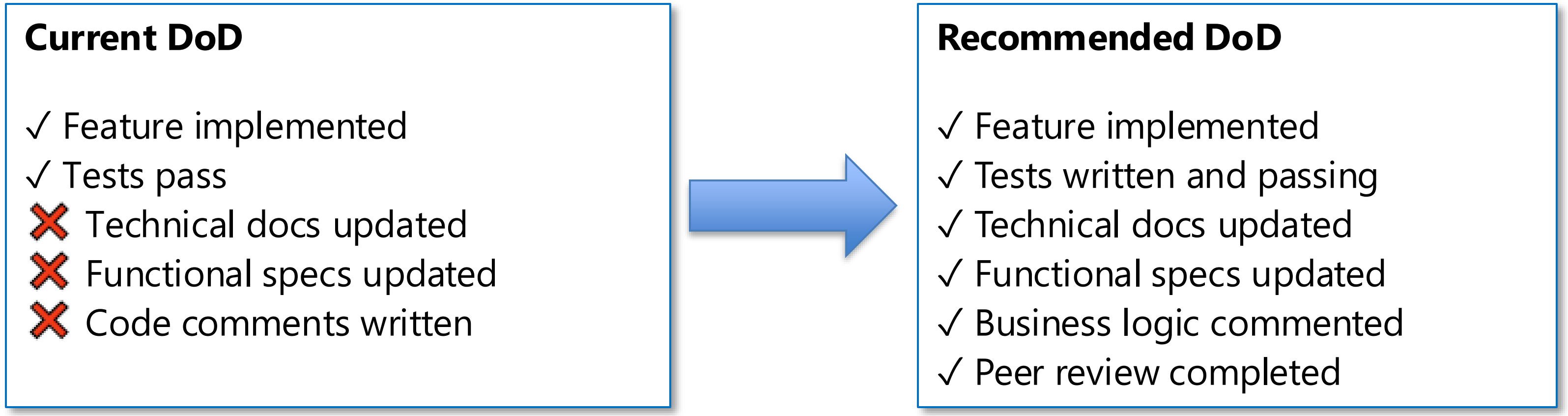
CISAC Experience
"Documentation is not available when we want it, when we ask for it.

We have to go back behind the supplier to be able to classify our documentation because it's all confusing."

Our Assessment (Oct 30):

☑ Documentation extensive (100+ pages of specifications)
⚠ Disorganized and difficult to navigate
⚠ Currency unclear (last modified 2019-2020)
☑ LLM-assisted reconciliation: Minimal drift vs .NET 3.1 code

Root Cause: Definition of Done Gap



Current DoD

✓ Feature implemented
✓ Tests pass
☒ Technical docs updated
☒ Functional specs updated
☒ Code comments written

Recommended DoD

✓ Feature implemented
✓ Tests written and passing
✓ Technical docs updated
✓ Functional specs updated
✓ Business logic commented
✓ Peer review completed

Transparency Issues – Pattern of Defensive Posture

Partnership Attitude Defensive Posture

"Cold atmosphere"
"I did audits in high privacy environment, people saw us as partners and were collaborative and enthusiastic"
(Guillaume's impression, Workshop 1)

"When you ask for something, it's always 'for what purpose'"
(Yann's experience)

❌ Access Requests

- Systematic questions about purpose
- Multiple objections raised sequentially
- Information shared reluctantly

❌ Git History

- "Source code provided, history is internal working process"
- 3+ weeks compliance review (still pending)

❌ Cost Correlation

- No automated tooling to explain monthly spending variations
- Manual investigation required via support tickets
- Logs kept only 3 months

❌ Performance Metrics

- No proactive dashboard sharing production metrics
- Relies on Spanish Point claims rather than shared objective data
- No formal SLAs for performance

PART 5

Vendor Lock-in Reality

Three Lock-in Mechanisms

Where Vendor Independence is Locked

🔴 **CRITICAL: Matching Engine**

- Proprietary "heart of the product" (contract restriction)
- Alternative vendors unknown

🔴 **CRITICAL: Infrastructure-as-Code**

- Smart AIM library proprietary (not in code delivery)
- 343 Azure resources to reverse-engineer

🟠 **HIGH RISK: Knowledge Transfer**

- Viability UNKNOWN - never tested with independent vendor
- 76,000+ lines, minimal documentation, high implicit knowledge

1. Matching Engine Deep-Dive - "Heart of the Product"

Aspect	Status	Details
✅ **Separation**	POSITIVE	Clean architectural separation, physically deployed separately
⚠️ **Performance**	CONCERN	(Probable) Synchronous blocking calls create performance coupling
🔴 **Access**	BLOCKER	Contractual restriction - source code inaccessible

📋 Contractual Constraints

- Source code access: Contract termination only
- Spanish Point position: "Heart of the product"
- Alternative vendors: Research needed (→ Phase 1)
- Licensing implications: Unknown

⚙️ Technical Coupling Details

- REST API integration: Synchronous HTTP calls
- Performance risk: Blocking calls may cause latency
- Alternative pattern: Queue-based async would decouple
- Replacement effort: Substantial refactoring required

2. What We Can't Access - Visibility Gaps

Resource	Status	Impact on Vendor Independence
✅ **ISWC Application Source Code** (.NET 8)	**GRANTED**	Can analyze architecture, code quality, integration patterns
⚠️ **Git Commit History**	**PENDING** (3+ weeks)	Cannot analyze evolution, developer turnover, knowledge concentration (bus factor)
🔴 **Matching Engine Source Code**	**BLOCKED** (Contract)	Cannot assess replacement feasibility, build alternative, or evaluate IP constraints
🔴 **IaC Templates & CI/CD Pipeline**	**EXCLUDED** (Proprietary)	Cannot reproduce 343 Azure resources, understand deployment process, enable new vendor
🟡 **Azure DevOps Board** (Task Mgmt)	**GRANTED** (Past week)	Late and reduced visibility for velocity, sprint planning, agile maturity
🟡 **Production Performance Metrics**	**LIMITED**	Relies on Spanish Point claims rather than shared dashboards
🟡 **Cost Correlation Data**	**MANUAL**	No automated tooling, support ticket required

3. Knowledge Transfer - The Critical Unknown

Factor	Reality	Impact on New Vendor
🔴 **Codebase Complexity**	76,000+ lines of code Significant duplication "Tentacular" dependencies	High cognitive load to understand Risk of introducing bugs during changes
🔴 **Documentation Gaps**	Minimal code comments No onboarding guide No architecture decision records (ADRs)	New developers face steep learning curve Cannot understand "why" decisions were made
🔴 **Implicit Knowledge**	Business logic reasoning undocumented Integration patterns learned over time Tribal knowledge from original developers	Critical context exists only in Spanish Point team's heads Lost if team changes or vendor switches
⚠️ **Unverified Assumption**	Never tested with independent vendor Spanish Point's own assessment: onboarding is hard No independent vendor has ever attempted transfer	**Cannot confirm vendor switch is even feasible** HIGHEST RISK

Vendor Switch Effort Estimate

Component	Effort (T-shirt)	Dependencies & Risks
Application Code Handover	M	• Knowledge transfer viability **UNKNOWN** • Minimal documentation • No onboarding process • **Recommend:** Pilot test first
IaC Reconstruction	S	• 343 Azure resources • Reverse-engineer from portal • **OR** license Smart AIM library (cost unknown)
Matching Engine Replacement	L (if required)	• Alternative vendors **UNKNOWN** • Substantial refactoring likely • **OR** accept ongoing lock-in
Knowledge Transfer (Overlap Period)	L (parallel run)	• Parallel vendor overlap recommended • Gradual transition reduces risk

HIGH uncertainty on **TIMELINE and COSTS** due to unknowns

PART 6

Strategic Recommendations

Our Strategic Approach

Test viability BEFORE committing to any path

THREE-PHASE APPROACH
Test → Decide → Execute

🧪 PHASE 1: Gather Data

- Knowledge transfer pilot test
- Matching engine market research
- Contract renegotiation (parallel)

🎯 PHASE 2: Decide Based on Results

- IF pilot succeeds → Path A (vendor switch)
- IF pilot fails → Path B (improved relationship)
- IF mixed results → Path C (partial independence)

🚀 PHASE 3: Execute Strategy

- Path A: XL timeline (full switch)
- Path B: Immediate (better terms)
- Path C: L timeline (partial)

KEY PRINCIPLE:
Don't decide today which path - decide when you have data

Phase 1 – Gather Data

#	Action	Details	Investment
1	**Knowledge Transfer Pilot Test**	• Assign small feature to independent vendor • Test maintainability with available materials • Assess Spanish Point handover cooperation	S - M
2	**Matching Engine Market Research**	• Identify alternative matching engine vendors • Request proposals/demos • Assess replacement feasibility	XS
3	**Contract Renegotiation (Parallel)**	→ See slides 35-36 for priority terms and details	Unknown

Success Criteria:

✓ Pilot test completed with clear feasibility assessment

✓ Market research identifies viable alternatives (or confirms none exist)

✓ Contract negotiation yields improved terms (at minimum: git history, IaC templates)

Phase 2-3 – Decision Framework & Execution Paths

Scenario	Condition	Path	Execution
✅ **SUCCESS**	**Pilot successful + Alternatives exist**	Path A: Vendor Transition	• XL timeline (12-24 months) • XL cost (€300-600K) • Full RFP process • Parallel vendor overlap • Gradual feature transition
🔴 **BLOCKED**	**Pilot fails OR No alternatives**	Path B: Improved Relationship	• Immediate • Minimal cost • Enhanced contract terms • Better transparency and governance • Cost optimization implemented • Continue with Spanish Point under better terms
⚠ **MIXED**	**Partial success Mixed results**	Path C: Partial Independence	• L timeline (6-12 months) • L cost • Keep Spanish Point for Matching Engine • New vendor for application maintenance • Clear interface contracts • Improved knowledge management

Key Principle: Based on Phase 1 results, choose strategic path

Phase 1 - The Knowledge Transfer Test

Scope: Small, Low-Risk Feature Assignment

Pilot projects examples

- Create a new environment, entirely managed outside of Spanish Point
- Databricks migration to new version or infrastructure
- Prototype of better efficiency for Cosmos DB usage and other coding optimizations

Selection Criteria

- Well-defined requirements
- Limited scope (XS effort for experienced team)
- Non-critical (failure doesn't impact production)
- Representative complexity (touches multiple layers)

Success Criteria

- Feature Delivered Correctly
- Reasonable Timeline
- Independent Delivery
- Code Quality Maintained
- Documentation Updated

Evaluation Questions

- How long to become productive?
- What knowledge gaps exist?
- Cooperation : Helpful or defensive?
- Code comprehension : Can they navigate codebase?
- Professional vs struggling Delivery Quality?

Phase 1 - Contract Renegotiation

CISAC's Strategic Asset: Data Authority

Priority	Terms	Details
🔴 **IMMEDIATE** (Must Have)	• Git access / synchro ability • Knowledge transfer improvements • Configuration files templates	No more compliance delays In source control delivery Automated dashboard
🟠 **HIGH PRIORITY** (Should have)	• IaC templates inclusion • Enhanced Definition of Done • Proactive metrics sharing • Monthly cost review meetings	Docs required Monthly dashboards Usage correlation
🟡 **MEDIUM PRIORITY** (Could have)	• Performance SLAs • Cost correlation tooling • Architecture Decision Records • Reduced environment setup costs • Smart AIM library licensing	Onboarding docs ADR practice IaC automation If vendor switch

Technical Quick Wins

Immediate Improvements

🔴 HIGH IMPACT, LOW EFFORT (Do Immediately)

Quick Win	Status	Impact	Effort	Action
Databricks Upgrade Planning	Version 10.4 LTS outdated	Missing features potential security gaps	Spanish Point upgrade roadmap (XS effort)	Request upgrade plan
Monthly Cost Review Meeting	No regular cost review process	€600K/year spending without visibility	Recurring meeting setup (1 hour/month)	Establish Meeting

🟠 HIGH IMPACT, MEDIUM EFFORT (Plan Next)

Quick Win	Status	Impact	Effort	Action
Cost Correlation Tooling	Manual investigation required	Cannot explain monthly variations	S	Either Spanish Point delivers OR CISAC builds using API Mgmt + Cost Mgmt APIs
Cosmos DB Archive Policy	Old audit logs in expensive Cosmos DB	Potential cost savings (estimated)	S	Archive 90+ day logs to Blob Storage
Performance Validation	Relying on Spanish Point claims	Cannot validate Hyperscale proposal need	XS	Gather production metrics user complaints

PART 7

Decision Points & Next Steps

Aspect	Details
Investment	S
Timeline	During Phase 1
Risk	Very Low (non-critical feature)
RECOMMENDATION	✅ **YES (High Priority)**
Rationale	Single most important data point for vendor independence strategy. Small investment de-risks XL decision.
Action Required	• Define pilot feature scope • Identify independent vendors to approach • Allocate budget approval

✅ DECISION 1: Approve Knowledge Transfer Pilot Test?

Strategic Decisions Required

Aspect	Details
Investment	Minimal (legal/negotiation time)
Timeline	During Phase 1
Risk	Low (worst case: status quo continues)
RECOMMENDATION	✅ **YES (High Priority)**
Rationale	Use audit findings as leverage. Better terms achievable whether CISAC stays or switches vendors.
Priority Terms	• **CRITICAL:** Git history, IaC templates, cost tooling • **HIGH:** Performance SLAs, enhanced DoD, metrics sharing • **MEDIUM:** Knowledge transfer docs, ADR practice
Action Required	• Yann to lead negotiation • Define priority vs negotiable terms

✅ DECISION 2: Initiate Contract Renegotiation?

Strategic Decisions Required

Aspect	Details
Investment	€40K migration - €1,800/month savings (Spanish Point proposal)
Timeline	Migration S after approval
Risk	Medium (reserved instance = 1-year commitment)
RECOMMENDATION	⚠ **HOLD Pending Validation**
Rationale	Bottleneck not validated. Query optimization may achieve similar results at lower cost. Secondary priority vs governance issues.
Actions BEFORE Approval	• Meet with Moaiz (production performance data) • Validate SQL Server bottleneck • Explore query optimization alternative (XS-S cost)

⚠ DECISION 3: Approve Hyperscale Proposal (PSA 5499)?

Strategic Decisions Required

Aspect	Details
Investment	XS (market research, vendor proposals)
Timeline	During Phase 1
Risk	Very Low (research only, no commitment)
RECOMMENDATION	✅ **YES (Medium Priority)**
Rationale	Strengthens negotiating position even if CISAC never switches. Informs long-term strategy. Small cost for high strategic value.
Scope	• Identify alternative matching engine vendors • Request proposals and demos • Assess functionality comparison • Evaluate API compatibility and integration effort
Action Required	• Define matching engine requirements spec • Research music rights technology vendors

✅ DECISION 4: Research Alternative Matching Engine Vendors?

Strategic Decisions Required

Aspect	Details
Investment	XS (market research, vendor proposals)
Timeline	After phase 1 result
Risk	Depends on path chosen
RECOMMENDATION	⏸ DEFER Until Phase 1 Complete
Rationale	Don't decide today. Wait for: • Pilot test result • Market research results • Negotiation outcomes (ongoing)

Path	Condition	Details
Path A: Vendor Transition	IF: Pilot successful + Alternatives exist	XL timeline, XL cost, high risk, full independence
Path B: Improved Relationship	IF: Pilot fails OR no alternatives	Immediate, minimal cost, medium risk, better terms
Path C: Partial Independence	IF: Mixed results	Split model: new vendor (app) + Spanish Point (ME)

⏸ DECISION 5: Vendor Relationship Strategy Going Forward?

Strategic Decisions Required

Merci

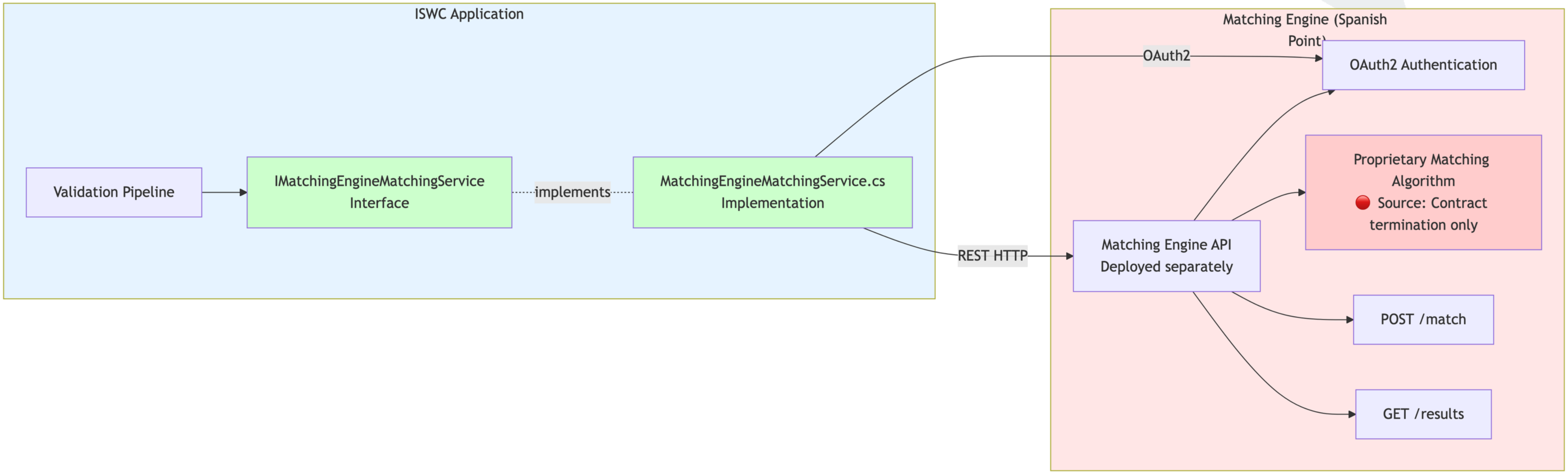
Questions et discussions

Bastien Gallay & Guillaume Jay

gallay@teragone-factory.com & jay@teragone-factory.com

ANNEX A

Detailed findings


ISWC Application
Validation Pipeline
IMatchingEngineMatchingService Interface
implements
MatchingEngineMatchingService.cs Implementation
OAuth2
REST HTTP
Matching Engine (Spanish Point)
OAuth2 Authentication
Proprietary Matching Algorithm
🔴 Source: Contract termination only
Matching Engine API Deployed separately
POST /match
GET /results

1. Matching Engine Deep-Dive - "Heart of the Product"

Slide Hyperscale Proposal - HOLD Pending Validation (Secondary Priority)

- **Hyperscale Proposal Assessment (PSA 5499)**

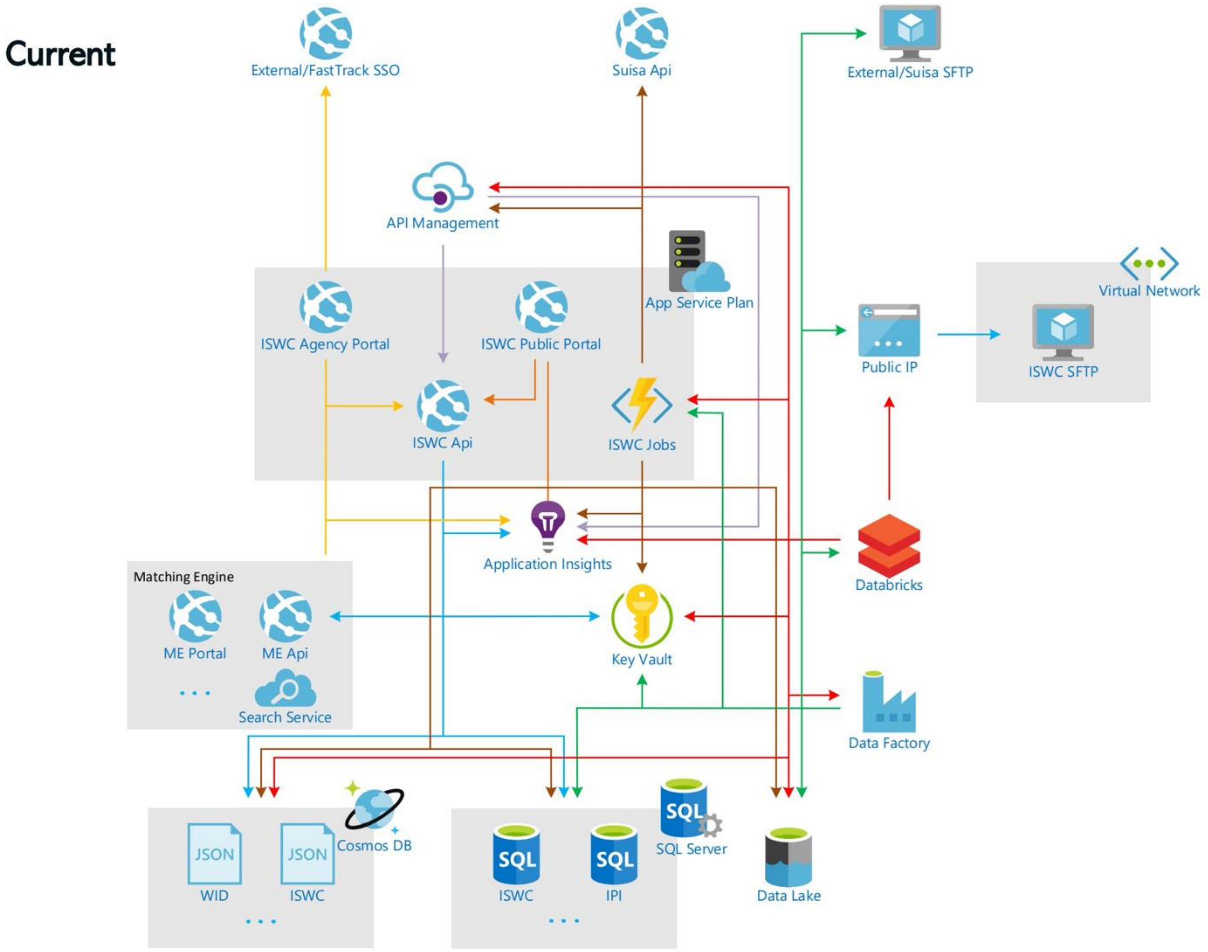
- **Three Objectives:**
- 1. Networking Security: Public → Private endpoints
- 2. SQL Tier Change: Business Critical → Hyperscale
- 3. Platform Modernization: Align with Microsoft best practices

- **Migration Investment: €40,000 (one-time)**
- **ROI: ~22 months payback period**

- **Our Assessment: ⚠ HOLD Pending Validation**

Cost Impact (Spanish Point Claims):

Component	Monthly Cost	Details
SQL Hyperscale Migration	**-€3,300/month**	Business Critical → Hyperscale + reserved instances
Private Networking	**+€1,500/month**	VPN Gateway Private DNS Private Endpoints WAF
NET SAVINGS	**-€1,800/month -€21,600/year**	



Azure Infrastructure

N+1 Issues : Best methods to convert to batch calling

IMatchingManager

- MatchAsync
- MatchIsrcsAsync
- RankMatches

IWorkManager

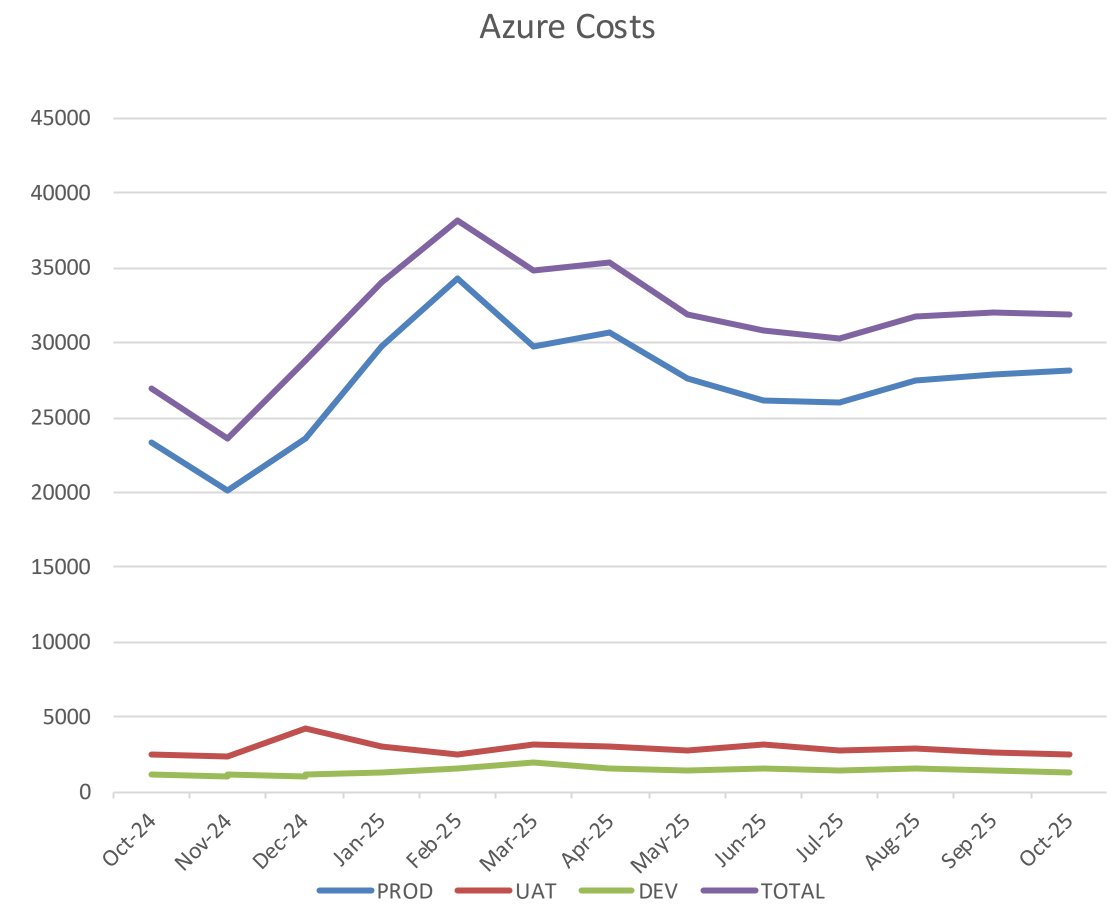
- GetCacheIswcs
- Exists
- FindIswcModelAsync
- CheckIfArchivedIswcAsync
- FindManyAsync (when called repeatedly in a Loop, despite its name)

IMessagingManager (minor impact)

- GetRejectionMessage

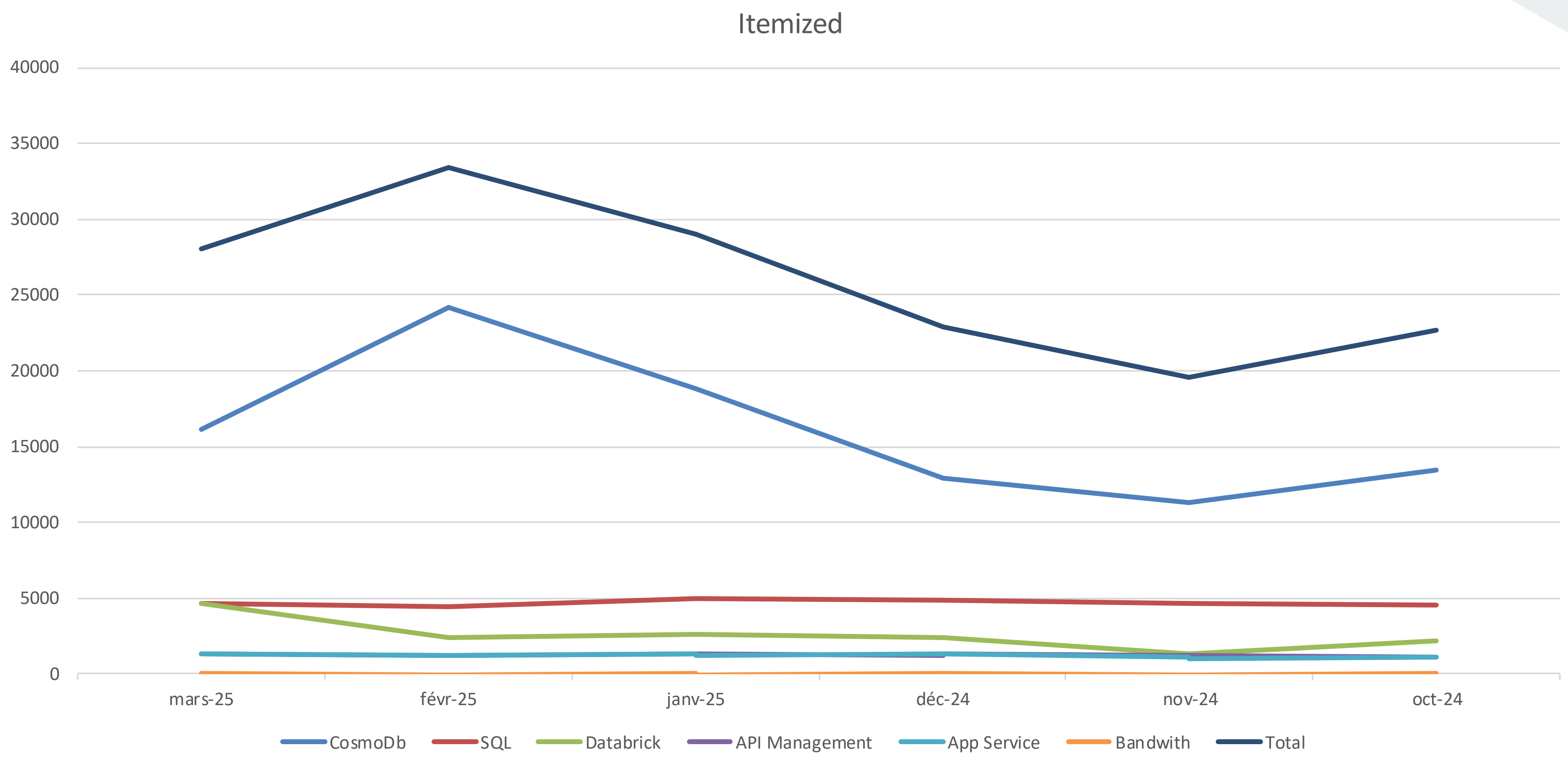
Workshops

2025-10-20	Kickoff
2025-10-21	Documentations and infrastructures
2025-11-05	Prod and Perf Data
2025-11-05	CI/CD Pipeline
2025-11-06	Cloud Cost Breakdown & Infrastructure Configuration
2025-11-10	Discussion Tech, Perf et Data
2025-11-10	Databricks
2025-11-12	Draft Restitution to Spanish Point

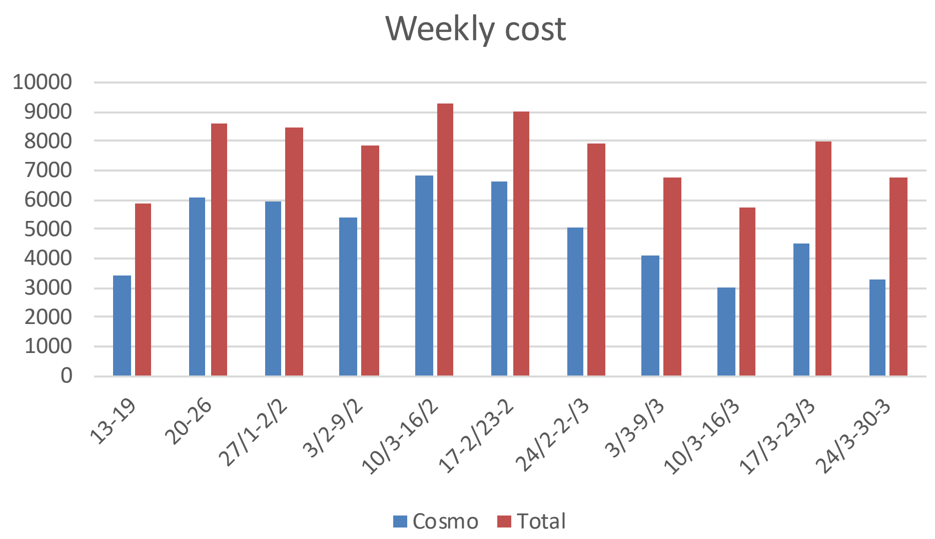


Feuille de calcul
Microsoft Excel

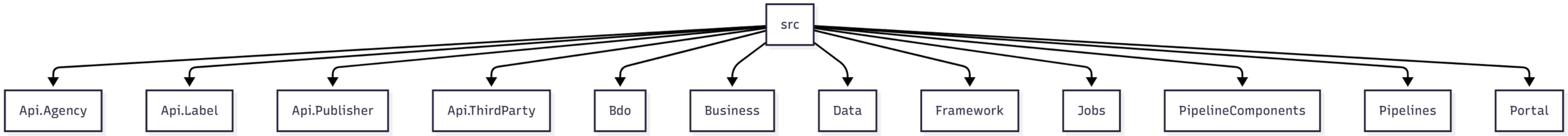
Costs : Total



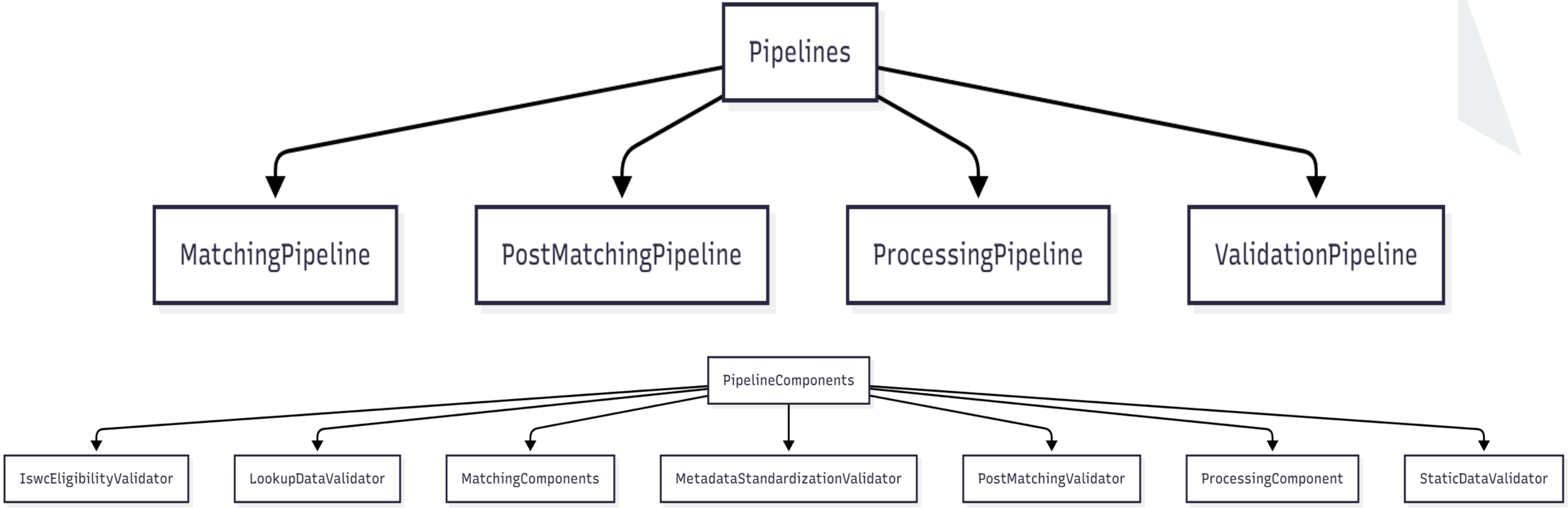
Costs : Itemized



Costs : Weekly Cost during Anomaly


src
Api.Agency
Api.Label
Api.Publisher
Api.ThirdParty
Bdo
Business
Data
Framework
Jobs
PipelineComponents
Pipelines
Portal

Code Tree



Sub Code Trees

ANNEX B

Validation Rules Reference

Overview

The ISWC Platform implements **95+ validation rules** across two pipeline stages:

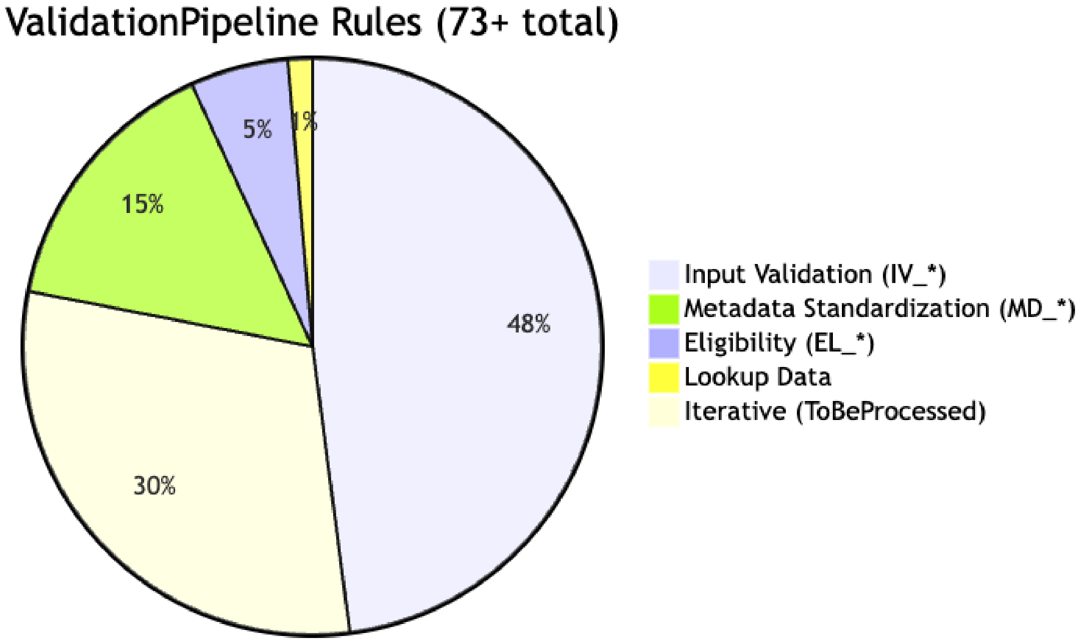
Pipeline Stage	Rule Count	Rule Prefixes	Error Code Range
ValidationPipeline	73+ rules	IV_, *MD_*, EL_, *PV_*	_100 to_126
PostMatchingPipeline	22 rules	PV_*, IV_40	_127 to_153, _247

Rule Types:

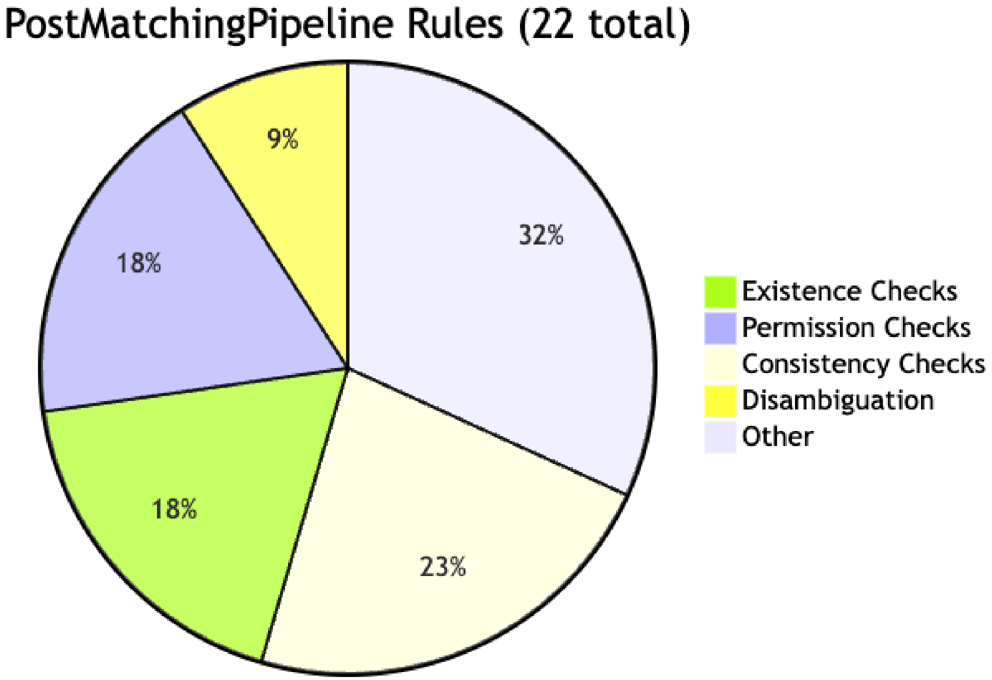
- **IRule**: Single-submission validation (most rules)
- **IBatchRule**: Batch-level validation (interface defined but **not implemented**)
- **IAlwaysOnRule**: Always-executed rules (cannot be disabled)
- **Configurable Rules**: Can be enabled/disabled via IRulesManager

Quick Reference: Rule Categories

ValidationPipeline (Pre-Processing)



PostMatchingPipeline (Post-Processing)



➡ More details in - *2025-11-24-CISAC-ISWC_Audit-validation-rules-catalog.pdf*

ANNEX C

Items Needing Verification

VERIFIED FACTS (High Confidence)

Finding	Evidence	Source
343 Azure resources deployed	Infrastructure inventory	Azure Portal
.NET 3.1 EOL Dec 2022, upgraded to 8.0 Nov 4	Microsoft docs + code analysis	Code Zip provided
700+ tests in CI/CD pipeline	Nov 5 workshop testimony	Workshop transcript
Matching Engine REST API integration	Code analysis from client-app (black box)	Source code
May-June 2024 production incident (6 months recovery)	Yann testimony	Oct 21 transcript, Line 41:40
CAB established May 2024	Yann testimony	Oct 21 transcript, Line 11:00
€50k/month cloud costs (€600K/year)	Yann statement	Oct 21 transcript, Line 22:23
No git history provided (zip format only)	Source code delivery format	Confirmed Nov 4
IaC templates not included (proprietary Smart AIM)	Nov 5 workshop	CI/CD workshop transcript
No automated cost correlation tooling	Spanish Point confirmation	Nov 6 workshop, Xiyuan statement

ASSUMPTIONS NEEDING VALIDATION (Medium-Low Confidence)

Assumption	Basis	Validation Needed	Priority
Knowledge transfer HIGH RISK	Code review observations	**CRITICAL: Pilot test required**	URGENT
Vendor switch timeline: 12-24 months	Audit team estimate	Requires detailed vendor proposals and scoping	HIGH
Vendor switch cost: €300-600K	Audit team educated guess	Requires detailed scoping, very low confidence	HIGH
Hyperscale savings: €3,300/month	Spanish Point PSA5499	Based on single month average; auto-scaling may vary	MEDIUM
ISWC API rate limit 3-5 req/sec	Yann Oct 30 statement	Not confirmed in Nov 5 workshop; need Moaiz validation	MEDIUM
Cost optimization potential: 10-20%	Audit team estimate (€60-120K/year)	Requires detailed Azure cost breakdown	MEDIUM
IaC reconstruction: 1-2 months	Audit team estimate	Technically feasible but effort uncertain	MEDIUM
20-day environment extension quote	Yann testimony	Need Spanish Point justification	MEDIUM
Matching Engine replacement effort: 6-12 months	Architecture analysis	Depends on alternatives available (unknown)	LOW

CODE ISSUES AND PATTERNS

Assumption	Basis	Validation Needed
N+1 Patterns	Code Analysis	Monitoring, Performance testing
Cosmos DB Efficiency Issues	Code review observations	A/B Testing, Monitoring
Cosmos DB Cache Usage	Code review observations	Comparison with recommended solutions