# **Ultimate AEC File Processing Architecture: Event-Driven Microservices with CQRS**

## **Executive Summary**

This architecture combines **Event-Driven Microservices**, **CQRS**, **Domain-Driven Design**, and **Reactive Patterns** to create the most scalable and user-friendly AEC file processing system. It handles millions of files across hundreds of projects while providing real-time insights and maintaining engineering workflow integrity.

## **Core Capabilities**

### **1. File Processing Capabilities**

* **Universal File Support**: CAD (.dwg, .dxf), BIM (.rvt, .ifc), PDFs, Office docs, images, point clouds
* **Intelligent Content Extraction**: Title blocks, specifications, dimensions, materials, equipment lists
* **Advanced OCR**: Technical drawing recognition with 95%+ accuracy
* **Semantic Analysis**: NLP-powered content understanding and relationship detection
* **Real-time Processing**: Sub-second file classification, minutes for full content extraction
* **Batch Processing**: Handle thousands of files simultaneously with backpressure management

### **2. AEC Domain Intelligence**

* **Revision Management**: Automatic supersession tracking with coordination analysis
* **Drawing Coordination**: Cross-discipline consistency checking (A-101 ↔ S-101 ↔ M-101)
* **Specification Integration**: Automatic linking between drawings and spec sections
* **Project Health Monitoring**: Real-time dashboards for project completeness and coordination
* **Standards Compliance**: Automated checking against company and industry standards
* **Deliverable Tracking**: Milestone-based delivery status across all disciplines

### **3. Integration Capabilities**

* **REST APIs**: Full CRUD operations with GraphQL support for complex queries
* **Event Streaming**: Real-time notifications via WebSockets, SignalR, or Kafka
* **Webhook Support**: Integrate with project management tools (Procore, PlanGrid, BIM 360)
* **File System Integration**: Direct integration with network drives, SharePoint, Box, Dropbox
* **ERP Integration**: Connect with Deltek, Sage, Oracle for project data synchronization

### **4. Analytics and Reporting**

* **Project Dashboards**: Real-time health metrics, coordination status, deliverable tracking
* **Predictive Analytics**: ML-powered project risk assessment and completion forecasting
* **Custom Reports**: Ad-hoc querying with natural language query interface
* **Data Export**: Support for PowerBI, Tableau, Excel, and custom visualization tools

## **System Architecture**

### **High-Level Architecture Diagram**

┌─────────────────────────────────────────────────────────────────────────────────┐

│ API Gateway │

│ (Authentication, Rate Limiting, Routing) │

└─────────────────────────────────────────────────────────────────────────────────┘

│

┌───────────────────────────────┼───────────────────────────────┐

│ │ │

┌────────▼─────────┐ ┌────────▼─────────┐ ┌────────▼─────────┐

│ Command API │ │ Query API │ │ Event Stream │

│ (Write Ops) │ │ (Read Ops) │ │ (Real-time) │

└────────┬─────────┘ └────────┬─────────┘ └────────┬─────────┘

│ │ │

│ │ │

┌────────▼─────────┐ ┌────────▼─────────┐ ┌────────▼─────────┐

│ Command Bus │ │ Read Model Store │ │ Event Bus │

│ │ │ (Materialized │ │ (Apache Kafka) │

│ │ │ Views) │ │ │

└────────┬─────────┘ └──────────────────┘ └────────┬─────────┘

│ │

│ │

┌────────▼──────────────────────────────────────────────────────────────▼─────────┐

│ Microservices Layer │

├─────────────────────┬─────────────────────┬─────────────────────┬───────────────┤

│ File Discovery │ Classification │ Content Processing │ Integration │

│ Service │ Service │ Services │ Services │

│ │ │ │ │

│ • File System │ • Naming Convention │ • CAD Processor │ • Webhook │

│ Monitoring │ Parser │ • PDF Processor │ Manager │

│ • Change Detection │ • Directory │ • BIM Processor │ • ERP Sync │

│ • Batch Discovery │ Classification │ • OCR Service │ • File Sync │

│ │ • Content Analysis │ • NLP Service │ │

└─────────────────────┴─────────────────────┴─────────────────────┴───────────────┘

│

│

┌────────────────────────────────────────▼─────────────────────────────────────────┐

│ Data Layer │

├─────────────────────┬─────────────────────┬─────────────────────┬───────────────┤

│ Event Store │ Data Lake │ Read Models │ Cache Layer │

│ (Event Sourcing) │ (Delta Lake) │ (PostgreSQL) │ (Redis) │

│ │ │ │ │

│ • Event History │ • File Records │ • Project Views │ • Hot Data │

│ • Audit Trail │ • Content Data │ • Coordination │ • Query Cache │

│ • Replay Capability │ • Binary Storage │ Matrix │ • Session │

│ │ • Analytics Data │ • Health Metrics │ Data │

└─────────────────────┴─────────────────────┴─────────────────────┴───────────────┘

## **Core Microservices**

### **1. File Discovery Service**

csharp

[Service("file-discovery")]

public class FileDiscoveryService : IFileDiscoveryService

{

private readonly IEventBus \_eventBus;

private readonly IFileSystemMonitor \_monitor;

private readonly IChangeDetectionService \_changeDetection;

public async Task StartMonitoringAsync(string projectPath)

{

await \_monitor.WatchDirectoryAsync(projectPath, async (fileEvent) =>

{

if (await ShouldProcessFileAsync(fileEvent))

{

await \_eventBus.PublishAsync(new FileDiscoveredEvent

{

FilePath = fileEvent.FullPath,

EventType = fileEvent.ChangeType,

ProjectRoot = projectPath,

DiscoveredAt = DateTime.UtcNow,

ProcessingId = Guid.NewGuid()

});

}

});

}

private async Task<bool> ShouldProcessFileAsync(FileSystemEventArgs fileEvent)

{

*// Intelligent filtering*

if (IsTemporaryFile(fileEvent.FullPath)) return false;

if (IsSystemFile(fileEvent.FullPath)) return false;

*// Change detection for efficiency*

var hasChanged = await \_changeDetection.HasFileChangedAsync(fileEvent.FullPath);

return hasChanged;

}

}

### **2. File Classification Service**

csharp

[Service("file-classification")]

public class FileClassificationService : IFileClassificationService

{

[EventHandler]

public async Task Handle(FileDiscoveredEvent @event)

{

var classification = await ClassifyFileAsync(@event.FilePath);

var directoryContext = AnalyzeDirectoryContext(@event.FilePath, @event.ProjectRoot);

await \_eventBus.PublishAsync(new FileClassifiedEvent

{

ProcessingId = @event.ProcessingId,

FilePath = @event.FilePath,

Classification = classification,

DirectoryContext = directoryContext,

ProcessingPriority = DetermineProcessingPriority(classification)

});

}

private async Task<AECFileClassification> ClassifyFileAsync(string filePath)

{

var fileName = Path.GetFileName(filePath);

var extension = Path.GetExtension(filePath).ToLower();

*// Parse standardized naming convention*

var parsedName = \_namingParser.Parse(fileName);

if (parsedName.IsStandardNaming)

{

return new AECFileClassification

{

ProjectNumber = parsedName.ProjectNumber,

Phase = MapPhase(parsedName.Phase),

Discipline = MapDiscipline(parsedName.Discipline),

DocumentType = MapDocumentType(parsedName.DocumentType),

SheetNumber = parsedName.SheetNumber,

RevisionNumber = parsedName.RevisionNumber,

RevisionType = MapRevisionType(parsedName.RevisionNumber),

ConfidenceLevel = 0.95f

};

}

*// Fallback to ML-based classification*

return await \_mlClassifier.ClassifyAsync(fileName, extension);

}

}

### **3. Content Processing Services**

#### **CAD Processing Service**

csharp

[Service("cad-processor")]

public class CADProcessingService : IContentProcessingService

{

[EventHandler]

public async Task Handle(FileClassifiedEvent @event)

{

if (!ShouldProcessCADFile(@event.Classification))

return;

var metadata = await ExtractCADMetadataAsync(@event.FilePath);

await \_eventBus.PublishAsync(new ContentExtractedEvent

{

ProcessingId = @event.ProcessingId,

FilePath = @event.FilePath,

ContentType = "CAD",

ExtractedMetadata = metadata,

ProcessingDuration = metadata.ProcessingTime

});

}

private async Task<CADMetadata> ExtractCADMetadataAsync(string filePath)

{

using var scope = \_serviceProvider.CreateScope();

var extractor = scope.ServiceProvider.GetRequiredService<ICADMetadataExtractor>();

var stopwatch = Stopwatch.StartNew();

var metadata = new CADMetadata();

try

{

using var database = new Database(false, true);

database.ReadDwgFile(filePath, FileShare.Read, true, "");

using var transaction = database.TransactionManager.StartTransaction();

*// Extract comprehensive metadata*

metadata.TitleBlock = await ExtractTitleBlockAsync(database, transaction);

metadata.TextEntities = await ExtractTextEntitiesAsync(database, transaction);

metadata.Dimensions = await ExtractDimensionsAsync(database, transaction);

metadata.Blocks = await ExtractBlocksAsync(database, transaction);

metadata.Layers = await ExtractLayersAsync(database, transaction);

metadata.DrawingBounds = ExtractDrawingBounds(database);

metadata.DrawingScale = ExtractDrawingScale(database);

metadata.DrawingUnits = ExtractDrawingUnits(database);

transaction.Commit();

}

catch (Exception ex)

{

metadata.ProcessingErrors.Add(ex.Message);

}

metadata.ProcessingTime = stopwatch.Elapsed;

return metadata;

}

}

#### **PDF Processing Service with Advanced OCR**

csharp

[Service("pdf-processor")]

public class PDFProcessingService : IContentProcessingService

{

private readonly IOCRService \_ocrService;

private readonly IPDFTextExtractor \_textExtractor;

private readonly IImagePreprocessor \_imagePreprocessor;

[EventHandler]

public async Task Handle(FileClassifiedEvent @event)

{

if (!ShouldProcessPDFFile(@event.Classification))

return;

var metadata = await ExtractPDFContentAsync(@event.FilePath, @event.Classification);

await \_eventBus.PublishAsync(new ContentExtractedEvent

{

ProcessingId = @event.ProcessingId,

FilePath = @event.FilePath,

ContentType = "PDF",

ExtractedMetadata = metadata

});

}

private async Task<PDFMetadata> ExtractPDFContentAsync(string filePath, AECFileClassification classification)

{

var metadata = new PDFMetadata();

*// Extract embedded text first (fast)*

var embeddedText = await \_textExtractor.ExtractTextAsync(filePath);

metadata.EmbeddedText = embeddedText;

*// Determine if OCR is needed*

var needsOCR = embeddedText.Length < 100 || IsScannedDrawing(embeddedText);

if (needsOCR && classification.DocumentType == DocumentType.Drawing)

{

*// Specialized OCR for technical drawings*

var ocrResult = await PerformTechnicalDrawingOCRAsync(filePath);

metadata.OCRText = ocrResult.ExtractedText;

metadata.OCRConfidence = ocrResult.AverageConfidence;

metadata.TechnicalElements = ocrResult.TechnicalElements;

}

else if (needsOCR)

{

*// Standard OCR for text documents*

var ocrResult = await \_ocrService.ExtractTextAsync(filePath);

metadata.OCRText = ocrResult.Text;

metadata.OCRConfidence = ocrResult.Confidence;

}

*// Extract structured information*

var combinedText = $"{embeddedText} {metadata.OCRText}";

metadata.TitleBlock = await ExtractTitleBlockFromTextAsync(combinedText);

metadata.TechnicalSpecifications = await ExtractTechnicalSpecsAsync(combinedText);

return metadata;

}

private async Task<TechnicalOCRResult> PerformTechnicalDrawingOCRAsync(string pdfPath)

{

var result = new TechnicalOCRResult();

*// Convert PDF to high-resolution images*

var images = await ConvertPDFToImagesAsync(pdfPath, dpi: 300);

foreach (var image in images)

{

*// Preprocess for technical drawings*

var processedImage = await \_imagePreprocessor.EnhanceForTechnicalContentAsync(image);

*// Multi-pass OCR with different engines*

var tesseractResult = await \_ocrService.ExtractWithTesseractAsync(processedImage,

new TesseractOptions

{

Language = "eng",

PageSegMode = PageSegmentationMode.SparseText,

EngineMode = EngineMode.NeuralNetsLSTMOnly,

Variables = new Dictionary<string, string>

{

["tessedit\_char\_whitelist"] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-.,/\"'()[]{}:;×°′″",

["classify\_enable\_learning"] = "0"

}

});

*// Specialized technical element detection*

var technicalElements = await DetectTechnicalElementsAsync(processedImage);

result.AddPageResult(tesseractResult, technicalElements);

}

return result;

}

}

### **4. Semantic Analysis Service**

csharp

[Service("semantic-analysis")]

public class SemanticAnalysisService : ISemanticAnalysisService

{

private readonly INLPProcessor \_nlpProcessor;

private readonly ITechnicalTerminologyService \_terminologyService;

private readonly IEntityExtractionService \_entityExtractor;

[EventHandler]

public async Task Handle(ContentExtractedEvent @event)

{

if (string.IsNullOrEmpty(@event.ExtractedMetadata.TextContent))

return;

var analysis = await PerformSemanticAnalysisAsync(

@event.ExtractedMetadata.TextContent,

@event.Classification);

await \_eventBus.PublishAsync(new SemanticAnalysisCompletedEvent

{

ProcessingId = @event.ProcessingId,

FilePath = @event.FilePath,

SemanticAnalysis = analysis

});

}

private async Task<SemanticAnalysis> PerformSemanticAnalysisAsync(string content, AECFileClassification classification)

{

var analysis = new SemanticAnalysis();

*// Extract named entities specific to AEC*

analysis.NamedEntities = await \_entityExtractor.ExtractAECEntitiesAsync(content, new[]

{

EntityType.Material,

EntityType.Equipment,

EntityType.Manufacturer,

EntityType.ProductModel,

EntityType.Dimension,

EntityType.Standard,

EntityType.Code,

EntityType.RoomSpace,

EntityType.System

});

*// Extract technical requirements and specifications*

analysis.TechnicalRequirements = await ExtractTechnicalRequirementsAsync(content, classification.Discipline);

*// Extract relationships between entities*

analysis.EntityRelationships = await \_nlpProcessor.ExtractRelationshipsAsync(

content, analysis.NamedEntities);

*// Content classification and intent detection*

analysis.ContentClassification = await ClassifyTechnicalContentAsync(content, classification);

*// Extract quantitative data*

analysis.Measurements = await ExtractMeasurementsAsync(content);

analysis.Quantities = await ExtractQuantitiesAsync(content);

return analysis;

}

private async Task<List<TechnicalRequirement>> ExtractTechnicalRequirementsAsync(string content, Discipline discipline)

{

var requirements = new List<TechnicalRequirement>();

*// Discipline-specific requirement patterns*

var patterns = GetRequirementPatterns(discipline);

foreach (var pattern in patterns)

{

var matches = pattern.Regex.Matches(content);

foreach (Match match in matches)

{

requirements.Add(new TechnicalRequirement

{

Type = pattern.Type,

Value = match.Groups["value"].Value,

Unit = match.Groups["unit"]?.Value,

Context = ExtractContext(content, match.Index, 200),

Confidence = CalculateConfidence(match, pattern),

Discipline = discipline

});

}

}

return requirements;

}

}

## **Data Architecture**

### **Event Store Schema**

csharp

public class EventStore

{

*// Event sourcing for complete audit trail*

public class Event

{

public Guid EventId { get; set; }

public string StreamId { get; set; } *// File processing ID*

public string EventType { get; set; }

public string EventData { get; set; } *// JSON payload*

public Dictionary<string, string> Metadata { get; set; }

public DateTime Timestamp { get; set; }

public int Version { get; set; }

}

*// Event streams by processing ID*

public async Task AppendEventsAsync(string streamId, IEnumerable<object> events)

{

var eventRecords = events.Select(e => new Event

{

EventId = Guid.NewGuid(),

StreamId = streamId,

EventType = e.GetType().Name,

EventData = JsonSerializer.Serialize(e),

Timestamp = DateTime.UtcNow,

Version = await GetNextVersionAsync(streamId)

});

await \_database.Events.AddRangeAsync(eventRecords);

await \_database.SaveChangesAsync();

*// Publish to event bus for real-time processing*

foreach (var evt in events)

{

await \_eventBus.PublishAsync(evt);

}

}

}

### **Data Lake Schema (Delta Lake)**

sql

*-- Core file records table with optimized partitioning*

CREATE TABLE aec\_file\_records (

*-- Primary identifiers*

file\_id STRING NOT NULL,

processing\_id STRING NOT NULL,

project\_number STRING NOT NULL,

*-- File metadata*

file\_path STRING NOT NULL,

file\_name STRING NOT NULL,

file\_size BIGINT,

file\_hash STRING,

mime\_type STRING,

*-- AEC Classification*

phase STRING NOT NULL,

discipline STRING NOT NULL,

document\_type STRING NOT NULL,

sheet\_number STRING,

revision\_number STRING,

revision\_type STRING,

is\_current\_revision BOOLEAN NOT NULL DEFAULT true,

*-- Timestamps*

file\_created\_date TIMESTAMP,

file\_modified\_date TIMESTAMP,

discovered\_date TIMESTAMP NOT NULL,

processed\_date TIMESTAMP,

last\_analyzed\_date TIMESTAMP,

*-- Processing status*

processing\_status STRING NOT NULL, *-- discovered, classified, processed, analyzed, completed, failed*

processing\_duration\_ms BIGINT,

error\_message STRING,

*-- Content indicators*

has\_embedded\_text BOOLEAN DEFAULT false,

has\_ocr\_text BOOLEAN DEFAULT false,

has\_title\_block BOOLEAN DEFAULT false,

has\_technical\_content BOOLEAN DEFAULT false,

content\_confidence\_score DOUBLE,

*-- Derived fields for optimization*

phase\_order INTEGER GENERATED ALWAYS AS (

CASE phase

WHEN 'PD' THEN 1 WHEN 'SD' THEN 2 WHEN 'DD' THEN 3

WHEN 'CD' THEN 4 WHEN 'CA' THEN 5 WHEN 'CO' THEN 6

ELSE 99 END),

discipline\_priority INTEGER GENERATED ALWAYS AS (

CASE discipline

WHEN 'A' THEN 1 WHEN 'S' THEN 2 WHEN 'M' THEN 3

WHEN 'E' THEN 4 WHEN 'P' THEN 5 WHEN 'F' THEN 6

ELSE 99 END),

*-- Partitioning columns*

year INTEGER GENERATED ALWAYS AS (YEAR(processed\_date)),

month INTEGER GENERATED ALWAYS AS (MONTH(processed\_date))

)

USING DELTA

PARTITIONED BY (project\_number, discipline, year)

CLUSTER BY (phase, document\_type, is\_current\_revision)

TBLPROPERTIES (

'delta.autoOptimize.optimizeWrite' = 'true',

'delta.autoOptimize.autoCompact' = 'true',

'delta.enableChangeDataFeed' = 'true',

'delta.deletedFileRetentionDuration' = 'interval 30 days'

);

*-- Extracted content table (separate for performance)*

CREATE TABLE aec\_file\_content (

content\_id STRING NOT NULL,

file\_id STRING NOT NULL,

content\_type STRING NOT NULL, *-- title\_block, embedded\_text, ocr\_text, technical\_specs, semantic\_analysis*

content\_data STRING, *-- JSON or compressed text*

content\_size BIGINT,

compression\_type STRING,

extraction\_method STRING,

confidence\_score DOUBLE,

created\_date TIMESTAMP NOT NULL,

*-- Content-specific metadata*

language STRING,

encoding STRING,

processing\_version STRING

)

USING DELTA

PARTITIONED BY (content\_type)

TBLPROPERTIES (

'delta.autoOptimize.optimizeWrite' = 'true'

);

*-- Technical entities and relationships*

CREATE TABLE aec\_technical\_entities (

entity\_id STRING NOT NULL,

file\_id STRING NOT NULL,

project\_number STRING NOT NULL,

entity\_type STRING NOT NULL, *-- material, equipment, manufacturer, standard, etc.*

entity\_value STRING NOT NULL,

entity\_category STRING,

confidence\_score DOUBLE,

context\_snippet STRING,

position\_in\_content INTEGER,

extraction\_method STRING,

created\_date TIMESTAMP NOT NULL

)

USING DELTA

PARTITIONED BY (project\_number, entity\_type);

*-- File relationships and coordination*

CREATE TABLE aec\_file\_relationships (

relationship\_id STRING NOT NULL,

source\_file\_id STRING NOT NULL,

target\_file\_id STRING NOT NULL,

relationship\_type STRING NOT NULL, *-- coordination, reference, supersession, dependency*

relationship\_strength DOUBLE, *-- 0.0 to 1.0*

relationship\_basis STRING, *-- sheet\_number, content\_similarity, explicit\_reference*

description STRING,

confidence\_score DOUBLE,

detected\_date TIMESTAMP NOT NULL,

verified\_date TIMESTAMP,

is\_verified BOOLEAN DEFAULT false,

project\_number STRING NOT NULL

)

USING DELTA

PARTITIONED BY (project\_number, relationship\_type);

### **Read Model Schemas (PostgreSQL)**

sql

*-- Optimized read models for common queries*

CREATE TABLE project\_current\_files\_view (

project\_number VARCHAR(50) NOT NULL,

discipline VARCHAR(10) NOT NULL,

phase VARCHAR(10) NOT NULL,

document\_type VARCHAR(20) NOT NULL,

total\_files INTEGER NOT NULL,

latest\_revision VARCHAR(20),

latest\_modified\_date TIMESTAMP,

coordination\_status VARCHAR(20), *-- synchronized, inconsistent, missing*

completeness\_score DECIMAL(5,2), *-- 0.00 to 100.00*

last\_updated TIMESTAMP NOT NULL DEFAULT NOW(),

PRIMARY KEY (project\_number, discipline, phase, document\_type)

);

CREATE INDEX idx\_project\_files\_project ON project\_current\_files\_view(project\_number);

CREATE INDEX idx\_project\_files\_discipline ON project\_current\_files\_view(discipline);

CREATE INDEX idx\_project\_files\_coordination ON project\_current\_files\_view(coordination\_status);

*-- Project health dashboard view*

CREATE TABLE project\_health\_dashboard\_view (

project\_number VARCHAR(50) PRIMARY KEY,

project\_name VARCHAR(200),

current\_phase VARCHAR(10),

overall\_health\_score DECIMAL(5,2), *-- 0.00 to 100.00*

*-- File statistics*

total\_files INTEGER NOT NULL DEFAULT 0,

current\_revision\_files INTEGER NOT NULL DEFAULT 0,

outdated\_files INTEGER NOT NULL DEFAULT 0,

*-- Discipline status*

disciplines\_active INTEGER NOT NULL DEFAULT 0,

disciplines\_current INTEGER NOT NULL DEFAULT 0,

*-- Coordination metrics*

coordination\_issues INTEGER NOT NULL DEFAULT 0,

critical\_coordination\_issues INTEGER NOT NULL DEFAULT 0,

*-- Processing metrics*

processing\_success\_rate DECIMAL(5,2),

average\_processing\_time\_ms BIGINT,

last\_processed\_file\_date TIMESTAMP,

*-- Timestamps*

last\_updated TIMESTAMP NOT NULL DEFAULT NOW(),

created\_date TIMESTAMP NOT NULL DEFAULT NOW()

);

*-- Coordination matrix view*

CREATE TABLE coordination\_matrix\_view (

project\_number VARCHAR(50) NOT NULL,

sheet\_base\_number VARCHAR(20) NOT NULL, *-- e.g., "101" from "A-101"*

architectural\_revision VARCHAR(20),

architectural\_date TIMESTAMP,

structural\_revision VARCHAR(20),

structural\_date TIMESTAMP,

mechanical\_revision VARCHAR(20),

mechanical\_date TIMESTAMP,

electrical\_revision VARCHAR(20),

electrical\_date TIMESTAMP,

plumbing\_revision VARCHAR(20),

plumbing\_date TIMESTAMP,

coordination\_status VARCHAR(20) NOT NULL, *-- synchronized, minor\_mismatch, major\_mismatch, missing\_disciplines*

mismatch\_severity INTEGER DEFAULT 0, *-- 0=none, 1=minor, 2=major, 3=critical*

last\_updated TIMESTAMP NOT NULL DEFAULT NOW(),

PRIMARY KEY (project\_number, sheet\_base\_number)

);

## **Implementation Strategy**

### **Phase 1: Core Infrastructure (Months 1-3)**

csharp

*// Infrastructure setup*

public class Phase1Implementation

{

*// 1. Set up event-driven infrastructure*

public async Task SetupEventInfrastructureAsync()

{

*// Apache Kafka for event streaming*

await SetupKafkaClusterAsync();

*// Event store (EventStore DB or custom)*

await SetupEventStoreAsync();

*// API Gateway (Ocelot or Kong)*

await SetupAPIGatewayAsync();

*// Service discovery (Consul or Eureka)*

await SetupServiceDiscoveryAsync();

*// Monitoring (Prometheus + Grafana)*

await SetupMonitoringAsync();

}

*// 2. Implement core services*

public async Task ImplementCoreServicesAsync()

{

*// File discovery service*

await DeployServiceAsync("file-discovery", new FileDiscoveryService());

*// File classification service*

await DeployServiceAsync("file-classification", new FileClassificationService());

*// Basic content processing*

await DeployServiceAsync("content-processing", new BasicContentProcessingService());

*// Data lake setup (Delta Lake on Azure/AWS)*

await SetupDataLakeAsync();

}

*// 3. Basic APIs*

public async Task ImplementBasicAPIsAsync()

{

*// Command API for file processing*

await DeployAPIAsync("command-api", new CommandAPI());

*// Query API for file retrieval*

await DeployAPIAsync("query-api", new QueryAPI());

*// Health monitoring endpoints*

await DeployAPIAsync("health-api", new HealthAPI());

}

}

### **Phase 2: Advanced Processing (Months 4-6)**

csharp

public class Phase2Implementation

{

*// 1. Specialized content processors*

public async Task ImplementSpecializedProcessorsAsync()

{

*// CAD processing with AutoCAD .NET API*

await DeployServiceAsync("cad-processor", new CADProcessingService());

*// PDF processing with advanced OCR*

await DeployServiceAsync("pdf-processor", new PDFProcessingService());

*// BIM processing (Revit API)*

await DeployServiceAsync("bim-processor", new BIMProcessingService());

*// Semantic analysis service*

await DeployServiceAsync("semantic-analysis", new SemanticAnalysisService());

}

*// 2. Advanced analytics*

public async Task ImplementAdvancedAnalyticsAsync()

{

*// Project health monitoring*

await DeployServiceAsync("health-monitor", new ProjectHealthMonitoringService());

*// Coordination analysis*

await DeployServiceAsync("coordination-analyzer", new CoordinationAnalysisService());

*// Predictive analytics*

await DeployServiceAsync("predictive-analytics", new PredictiveAnalyticsService());

}

}

### **Phase 3: Integration & UI (Months 7-9)**

csharp

public class Phase3Implementation

{

*// 1. External integrations*

public async Task ImplementIntegrationsAsync()

{

*// Webhook management*

await DeployServiceAsync("webhook-manager", new WebhookManagementService());

*// ERP integrations (Deltek, Sage)*

await DeployServiceAsync("erp-integration", new ERPIntegrationService());

*// File system integrations*

await DeployServiceAsync("file-sync", new FileSynchronizationService());

}

*// 2. User interfaces*

public asyScalability Features

1. Horizontal Scaling

yaml# Kubernetes deployment configuration

apiVersion: apps/v1RetryNContinueEdityaml# Kubernetes deployment configuration

apiVersion: apps/v1

kind: Deployment

metadata:

name: cad-processor

spec:

replicas: 5 # Auto-scaling based on queue depth

selector:

matchLabels:

app: cad-processor

template:

metadata:

labels:

app: cad-processor

spec:

containers:

- name: cad-processor

image: aec-system/cad-processor:latest

resources:

requests:

memory: "2Gi"

cpu: "1000m"

limits:

memory: "8Gi"

cpu: "4000m"

env:

- name: KAFKA\_BROKERS

value: "kafka-cluster:9092"

- name: DATA\_LAKE\_CONNECTION

valueFrom:

secretKeyRef:

name: data-lake-secret

key: connection-string

---

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: cad-processor-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: cad-processor

minReplicas: 2

maxReplicas: 20

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 70

- type: External

external:

metric:

name: kafka\_consumer\_lag

target:

type: AverageValue

averageValue: "100" # Scale up if queue lag > 100 messages

2. Intelligent Load Balancing

csharppublic class IntelligentLoadBalancer : ILoadBalancer

{

private readonly IServiceDiscovery \_serviceDiscovery;

private readonly IMetricsCollector \_metrics;

private readonly Dictionary<string, ServiceCapability> \_serviceCapabilities;

public async Task<ServiceEndpoint> SelectServiceAsync(ProcessingRequest request)

{

var availableServices = await \_serviceDiscovery.GetHealthyServicesAsync(request.ServiceType);

// Multi-factor load balancing based on:

// 1. Current CPU/Memory usage

// 2. Queue depth

// 3. File type specialization

// 4. Processing history for similar files

var scoredServices = new List<(ServiceEndpoint endpoint, double score)>();

foreach (var service in availableServices)

{

var metrics = await \_metrics.GetCurrentMetricsAsync(service.Id);

var capability = \_serviceCapabilities[service.Id];

var score = CalculateServiceScore(request, metrics, capability);

scoredServices.Add((service, score));

}

// Select service with highest score

var selectedService = scoredServices.OrderByDescending(s => s.score).First();

// Update routing history for learning

await UpdateRoutingHistoryAsync(request, selectedService.endpoint);

return selectedService.endpoint;

}

private double CalculateServiceScore(ProcessingRequest request, ServiceMetrics metrics, ServiceCapability capability)

{

double score = 1.0;

// CPU/Memory availability (0.0 to 1.0)

var resourceScore = (1.0 - metrics.CpuUsage) \* 0.3 + (1.0 - metrics.MemoryUsage) \* 0.2;

// Queue depth penalty

var queueScore = Math.Max(0.0, 1.0 - (metrics.QueueDepth / 1000.0)) \* 0.2;

// File type specialization bonus

var specializationScore = capability.GetSpecializationScore(request.FileType) \* 0.2;

// Historical performance bonus

var performanceScore = capability.GetHistoricalPerformanceScore(request.FileType, request.FileSize) \* 0.1;

return score \* (resourceScore + queueScore + specializationScore + performanceScore);

}

}

3. Adaptive Resource Management

csharppublic class AdaptiveResourceManager : IResourceManager

{

private readonly IClusterManager \_clusterManager;

private readonly IWorkloadPredictor \_workloadPredictor;

private readonly Timer \_optimizationTimer;

public AdaptiveResourceManager()

{

// Run optimization every 5 minutes

\_optimizationTimer = new Timer(OptimizeResources, null,

TimeSpan.FromMinutes(5), TimeSpan.FromMinutes(5));

}

private async void OptimizeResources(object state)

{

try

{

var currentWorkload = await AnalyzeCurrentWorkloadAsync();

var predictedWorkload = await \_workloadPredictor.PredictNextHourAsync();

var optimizations = DetermineOptimizations(currentWorkload, predictedWorkload);

foreach (var optimization in optimizations)

{

await ApplyOptimizationAsync(optimization);

}

}

catch (Exception ex)

{

\_logger.LogError(ex, "Error during resource optimization");

}

}

private List<ResourceOptimization> DetermineOptimizations(

WorkloadAnalysis current,

WorkloadPrediction predicted)

{

var optimizations = new List<ResourceOptimization>();

// Scale up if queue depth is growing

if (current.QueueDepth > 1000 && predicted.QueueGrowthRate > 0.1)

{

optimizations.Add(new ScaleUpOptimization

{

ServiceType = current.BottleneckService,

TargetReplicas = CalculateOptimalReplicas(current, predicted),

Priority = OptimizationPriority.High

});

}

// Scale down if resources are underutilized

if (current.AverageResourceUtilization < 0.3 && predicted.WorkloadTrend == WorkloadTrend.Decreasing)

{

optimizations.Add(new ScaleDownOptimization

{

ServiceType = current.OverprovisionedService,

TargetReplicas = Math.Max(2, current.CurrentReplicas - 2),

Priority = OptimizationPriority.Medium

});

}

// Redistribute workload if there's imbalance

if (current.LoadImbalanceRatio > 2.0)

{

optimizations.Add(new LoadRebalanceOptimization

{

SourceServices = current.OverloadedServices,

TargetServices = current.UnderutilizedServices,

Priority = OptimizationPriority.High

});

}

return optimizations;

}

}

User Experience Features

1. Real-Time Dashboard

typescript// React dashboard with real-time updates

import React, { useState, useEffect } from 'react';

import { SignalRConnectionBuilder } from '@microsoft/signalr';

interface ProjectHealthDashboard {

projectNumber: string;

overallHealth: number;

fileStatistics: FileStatistics;

coordinationStatus: CoordinationStatus;

recentActivity: ActivityItem[];

}

export const ProjectHealthDashboard: React.FC = () => {

const [dashboardData, setDashboardData] = useState<ProjectHealthDashboard>();

const [connection, setConnection] = useState<signalR.HubConnection>();

useEffect(() => {

// Establish SignalR connection for real-time updates

const newConnection = new SignalRConnectionBuilder()

.withUrl('/hubs/projectHealth')

.withAutomaticReconnect()

.build();

newConnection.start().then(() => {

console.log('Connected to project health hub');

// Subscribe to project updates

newConnection.on('ProjectHealthUpdated', (data: ProjectHealthDashboard) => {

setDashboardData(data);

// Show toast notification for significant changes

if (data.coordinationStatus.criticalIssues > 0) {

showNotification('Critical coordination issues detected!', 'error');

}

});

// Subscribe to file processing updates

newConnection.on('FileProcessed', (fileInfo: ProcessedFileInfo) => {

updateFileStatistics(fileInfo);

showNotification(`Processed: ${fileInfo.fileName}`, 'success');

});

});

setConnection(newConnection);

return () => {

newConnection?.stop();

};

}, []);

const handleRefreshProject = async () => {

try {

const response = await fetch(`/api/projects/${projectNumber}/refresh`, {

method: 'POST'

});

if (response.ok) {

showNotification('Project refresh initiated', 'info');

}

} catch (error) {

showNotification('Failed to refresh project', 'error');

}

};

return (

<div className="project-health-dashboard">

<div className="dashboard-header">

<h1>Project Health Dashboard</h1>

<div className="health-score">

<CircularProgress

value={dashboardData?.overallHealth || 0}

color={getHealthColor(dashboardData?.overallHealth || 0)}

/>

<span>Overall Health</span>

</div>

</div>

<div className="dashboard-grid">

<FileStatisticsCard stats={dashboardData?.fileStatistics} />

<CoordinationMatrixCard matrix={dashboardData?.coordinationStatus} />

<RecentActivityCard activities={dashboardData?.recentActivity} />

<ProcessingQueueCard />

</div>

<div className="dashboard-actions">

<button onClick={handleRefreshProject}>Refresh Project</button>

<button onClick={() => exportHealthReport()}>Export Report</button>

</div>

</div>

);

};

// Coordination Matrix Component with interactive features

export const CoordinationMatrixCard: React.FC<{matrix: CoordinationStatus}> = ({matrix}) => {

const [selectedSheet, setSelectedSheet] = useState<string>();

return (

<div className="coordination-matrix-card">

<h3>Coordination Matrix</h3>

<div className="matrix-grid">

{matrix?.sheets.map(sheet => (

<div

key={sheet.sheetNumber}

className={`matrix-cell ${getCoordinationStatusClass(sheet.status)}`}

onClick={() => setSelectedSheet(sheet.sheetNumber)}

>

<div className="sheet-number">{sheet.sheetNumber}</div>

<div className="revision-info">

<span className="arch">A: {sheet.architecturalRevision}</span>

<span className="struct">S: {sheet.structuralRevision}</span>

<span className="mech">M: {sheet.mechanicalRevision}</span>

<span className="elec">E: {sheet.electricalRevision}</span>

</div>

{sheet.issues.length > 0 && (

<div className="issue-indicator">

<ExclamationTriangleIcon />

<span>{sheet.issues.length}</span>

</div>

)}

</div>

))}

</div>

{selectedSheet && (

<SheetDetailsModal

sheetNumber={selectedSheet}

onClose={() => setSelectedSheet(undefined)}

/>

)}

</div>

);

};

2. Natural Language Query Interface

typescript// AI-powered query interface

export const NaturalLanguageQuery: React.FC = () => {

const [query, setQuery] = useState('');

const [results, setResults] = useState<QueryResult[]>([]);

const [isLoading, setIsLoading] = useState(false);

const handleQuery = async (naturalLanguageQuery: string) => {

setIsLoading(true);

try {

// Send to NLP service for query parsing

const response = await fetch('/api/query/natural', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ query: naturalLanguageQuery })

});

const results = await response.json();

setResults(results);

} catch (error) {

console.error('Query failed:', error);

} finally {

setIsLoading(false);

}

};

const exampleQueries = [

"Show me all current architectural drawings for Project ABC123",

"Find coordination issues between structural and mechanical drawings",

"List all specifications that reference steel beams",

"What files were processed in the last 24 hours?",

"Show me the revision history for sheet A-101",

"Find all CAD files larger than 50MB that haven't been updated in 30 days"

];

return (

<div className="natural-language-query">

<div className="query-input">

<input

type="text"

value={query}

onChange={(e) => setQuery(e.target.value)}

placeholder="Ask anything about your project files..."

onKeyPress={(e) => e.key === 'Enter' && handleQuery(query)}

/>

<button onClick={() => handleQuery(query)} disabled={isLoading}>

{isLoading ? <Spinner /> : <SearchIcon />}

</button>

</div>

<div className="example-queries">

<h4>Try these examples:</h4>

{exampleQueries.map((example, index) => (

<button

key={index}

className="example-query"

onClick={() => handleQuery(example)}

>

{example}

</button>

))}

</div>

<div className="query-results">

{results.map((result, index) => (

<QueryResultCard key={index} result={result} />

))}

</div>

</div>

);

};

// Natural language query processing service

public class NaturalLanguageQueryService : INaturalLanguageQueryService

{

private readonly IOpenAIService \_openAI;

private readonly IQueryExecutor \_queryExecutor;

private readonly IQueryCache \_cache;

public async Task<QueryResult[]> ProcessNaturalLanguageQueryAsync(string naturalLanguageQuery)

{

// Check cache first

var cacheKey = GenerateCacheKey(naturalLanguageQuery);

if (await \_cache.TryGetAsync(cacheKey, out QueryResult[] cachedResults))

{

return cachedResults;

}

// Parse natural language to structured query

var structuredQuery = await ParseToStructuredQueryAsync(naturalLanguageQuery);

// Execute structured query

var results = await \_queryExecutor.ExecuteAsync(structuredQuery);

// Cache results

await \_cache.SetAsync(cacheKey, results, TimeSpan.FromMinutes(30));

return results;

}

private async Task<StructuredQuery> ParseToStructuredQueryAsync(string naturalLanguageQuery)

{

var prompt = $@"

Convert this natural language query about AEC project files into a structured query:

Natural Language: ""{naturalLanguageQuery}""

Available fields:

- project\_number, file\_name, file\_path, discipline, phase, document\_type

- sheet\_number, revision\_number, is\_current\_revision

- file\_size, created\_date, modified\_date, processed\_date

- has\_coordination\_issues, coordination\_status

- content\_type, technical\_keywords

Respond with JSON in this format:

{{

""tables"": [""aec\_file\_records""],

""select"": [""field1"", ""field2""],

""where"": [

{{""field"": ""discipline"", ""operator"": ""="", ""value"": ""A""}},

{{""field"": ""is\_current\_revision"", ""operator"": ""="", ""value"": true}}

],

""orderBy"": [{{""field"": ""modified\_date"", ""direction"": ""desc""}}],

""limit"": 100,

""intent"": ""find\_current\_architectural\_files""

}}";

var response = await \_openAI.CompleteAsync(prompt, new OpenAIOptions

{

Model = "gpt-4",

Temperature = 0.1,

MaxTokens = 500

});

return JsonSerializer.Deserialize<StructuredQuery>(response.Text);

}

}

3. Advanced Search and Filtering

csharppublic class AdvancedSearchService : IAdvancedSearchService

{

private readonly IElasticsearchClient \_elasticClient;

private readonly IDataLakeRepository \_dataLake;

public async Task<SearchResults> SearchAsync(SearchRequest request)

{

// Multi-modal search combining:

// 1. Metadata search (structured data)

// 2. Full-text search (document content)

// 3. Semantic search (similar meaning)

// 4. Visual search (similar drawings)

var searchDescriptor = new SearchDescriptor<FileDocument>()

.Index("aec-files")

.Size(request.PageSize)

.From(request.PageSize \* request.PageNumber);

// Build complex query

var queryContainers = new List<QueryContainer>();

// Metadata filters

if (!string.IsNullOrEmpty(request.ProjectNumber))

{

queryContainers.Add(Query<FileDocument>.Term(f => f.ProjectNumber, request.ProjectNumber));

}

if (request.Disciplines?.Any() == true)

{

queryContainers.Add(Query<FileDocument>.Terms(f => f.Discipline, request.Disciplines));

}

if (request.DateRange != null)

{

queryContainers.Add(Query<FileDocument>.DateRange(dr => dr

.Field(f => f.ModifiedDate)

.GreaterThanOrEquals(request.DateRange.From)

.LessThanOrEquals(request.DateRange.To)));

}

// Full-text search

if (!string.IsNullOrEmpty(request.TextQuery))

{

queryContainers.Add(Query<FileDocument>.MultiMatch(mm => mm

.Fields(f => f

.Field(fd => fd.FileName, boost: 3.0)

.Field(fd => fd.ExtractedText, boost: 2.0)

.Field(fd => fd.TitleBlock.Title, boost: 2.5)

.Field(fd => fd.TechnicalKeywords, boost: 1.5))

.Query(request.TextQuery)

.Type(TextQueryType.BestFields)

.Fuzziness(Fuzziness.Auto)));

}

// Semantic search using vector embeddings

if (!string.IsNullOrEmpty(request.SemanticQuery))

{

var queryEmbedding = await GenerateEmbeddingAsync(request.SemanticQuery);

queryContainers.Add(Query<FileDocument>.Script(s => s

.Script(ss => ss

.Source($"cosineSimilarity(params.query\_vector, 'content\_embedding') + 1.0")

.Params(p => p.Add("query\_vector", queryEmbedding)))));

}

// Combine all queries

searchDescriptor.Query(q => q.Bool(b => b.Must(queryContainers.ToArray())));

// Add aggregations for faceted search

searchDescriptor.Aggregations(a => a

.Terms("disciplines", t => t.Field(f => f.Discipline))

.Terms("phases", t => t.Field(f => f.Phase))

.Terms("document\_types", t => t.Field(f => f.DocumentType))

.DateHistogram("modified\_dates", d => d

.Field(f => f.ModifiedDate)

.CalendarInterval(DateInterval.Month)));

// Add highlighting

searchDescriptor.Highlight(h => h

.Fields(f => f

.Field(fd => fd.ExtractedText)

.Field(fd => fd.FileName)

.Field(fd => fd.TitleBlock.Title))

.PreTags("<mark>")

.PostTags("</mark>"));

var response = await \_elasticClient.SearchAsync<FileDocument>(searchDescriptor);

return new SearchResults

{

Documents = response.Documents.ToList(),

TotalHits = response.Total,

Facets = ExtractFacets(response.Aggregations),

Highlights = ExtractHighlights(response.Hits),

QueryTime = response.Took

};

}

private async Task<float[]> GenerateEmbeddingAsync(string text)

{

// Use OpenAI embeddings or similar service

var response = await \_openAI.CreateEmbeddingAsync(new EmbeddingRequest

{

Input = text,

Model = "text-embedding-ada-002"

});

return response.Data[0].Embedding.ToArray();

}

}

Deployment and Operations

1. Infrastructure as Code

yaml# Terraform configuration for Azure deployment

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "~> 3.0"

}

}

}

provider "azurerm" {

features {}

}

# Resource Group

resource "azurerm\_resource\_group" "aec\_processing" {

name = "rg-aec-processing-${var.environment}"

location = var.location

}

# Azure Kubernetes Service

resource "azurerm\_kubernetes\_cluster" "aks" {

name = "aks-aec-processing-${var.environment}"

location = azurerm\_resource\_group.aec\_processing.location

resource\_group\_name = azurerm\_resource\_group.aec\_processing.name

dns\_prefix = "aec-processing-${var.environment}"

default\_node\_pool {

name = "default"

node\_count = 3

vm\_size = "Standard\_D4s\_v3"

auto\_scaling\_enabled = true

min\_count = 3

max\_count = 20

}

# Specialized node pool for CAD processing

dynamic "node\_pool" {

for\_each = var.enable\_cad\_processing ? [1] : []

content {

name = "cadprocessing"

node\_count = 2

vm\_size = "Standard\_D8s\_v3"

node\_taints = ["workload=cad-processing:NoSchedule"]

}

}

identity {

type = "SystemAssigned"

}

}

# Azure Data Lake Storage Gen2

resource "azurerm\_storage\_account" "data\_lake" {

name = "adlsaecprocessing${var.environment}"

resource\_group\_name = azurerm\_resource\_group.aec\_processing.name

location = azurerm\_resource\_group.aec\_processing.location

account\_tier = "Standard"

account\_replication\_type = "ZRS"

is\_hns\_enabled = true # Hierarchical namespace for Data Lake

blob\_properties {

versioning\_enabled = true

delete\_retention\_policy {

days = 30

}

}

}

# Event Hubs for event streaming

resource "azurerm\_eventhub\_namespace" "event\_hub" {

name = "evhns-aec-processing-${var.environment}"

location = azurerm\_resource\_group.aec\_processing.location

resource\_group\_name = azurerm\_resource\_group.aec\_processing.name

sku = "Standard"

capacity = 2

auto\_inflate\_enabled = true

maximum\_throughput\_units = 20

}

resource "azurerm\_eventhub" "file\_events" {

name = "file-processing-events"

namespace\_name = azurerm\_eventhub\_namespace.event\_hub.name

resource\_group\_name = azurerm\_resource\_group.aec\_processing.name

partition\_count = 8

message\_retention = 7

}

# Azure Database for PostgreSQL (Read models)

resource "azurerm\_postgresql\_flexible\_server" "read\_models" {

name = "psql-aec-readmodels-${var.environment}"

resource\_group\_name = azurerm\_resource\_group.aec\_processing.name

location = azurerm\_resource\_group.aec\_processing.location

version = "14"

administrator\_login = var.db\_admin\_username

administrator\_password = var.db\_admin\_password

zone = "1"

storage\_mb = 32768

sku\_name = "GP\_Standard\_D4s\_v3"

backup\_retention\_days = 7

geo\_redundant\_backup\_enabled = true

}

# Azure Redis Cache

resource "azurerm\_redis\_cache" "cache" {

name = "redis-aec-cache-${var.environment}"

location = azurerm\_resource\_group.aec\_processing.location

resource\_group\_name = azurerm\_resource\_group.aec\_processing.name

capacity = 2

family = "C"

sku\_name = "Standard"

enable\_non\_ssl\_port = false

minimum\_tls\_version = "1.2"

}

# Application Insights

resource "azurerm\_application\_insights" "app\_insights" {

name = "appi-aec-processing-${var.environment}"

location = azurerm\_resource\_group.aec\_processing.location

resource\_group\_name = azurerm\_resource\_group.aec\_processing.name

application\_type = "web"

}

2. Monitoring and Observability

csharppublic class ComprehensiveMonitoring

{

// Application Performance Monitoring

public class APMConfiguration

{

public static void ConfigureAPM(IServiceCollection services)

{

// Application Insights

services.AddApplicationInsightsTelemetry();

// Custom metrics

services.AddSingleton<IMetricsCollector, MetricsCollector>();

// Distributed tracing

services.AddOpenTelemetry()

.WithTracing(builder => builder

.AddAspNetCoreInstrumentation()

.AddHttpClientInstrumentation()

.AddSqlClientInstrumentation()

.AddJaegerExporter());

// Health checks

services.AddHealthChecks()

.AddCheck<DatabaseHealthCheck>("database")

.AddCheck<EventBusHealthCheck>("eventbus")

.AddCheck<DataLakeHealthCheck>("datalake")

.AddCheck<FileSystemHealthCheck>("filesystem");

}

}

// Business Metrics Collection

public class BusinessMetricsCollector : IBusinessMetricsCollector

{

private readonly IMetricsLogger \_metrics;

public async Task RecordFileProcessingMetrics(FileProcessingResult result)

{

// Processing time by file type

\_metrics.RecordValue("file\_processing\_duration\_ms",

result.ProcessingDuration.TotalMilliseconds,

new[] {

KeyValuePair.Create("file\_type", result.FileType),

KeyValuePair.Create("file\_size\_category", GetSizeCategory(result.FileSize)),

KeyValuePair.Create("status", result.Status.ToString())

});

// Throughput metrics

\_metrics.IncrementCounter("files\_processed\_total",

new[] { KeyValuePair.Create("discipline", result.Discipline) });

// Quality metrics

if (result.ContentExtractionConfidence.HasValue)

{

\_metrics.RecordValue("content\_extraction\_confidence",

result.ContentExtractionConfidence.Value,

new[] { KeyValuePair.Create("extraction\_method", result.ExtractionMethod) });

}

// Error tracking

if (result.HasErrors)

{

\_metrics.IncrementCounter("processing\_errors\_total",

new[] {

KeyValuePair.Create("error\_type", result.ErrorType),

KeyValuePair.Create("file\_type", result.FileType)

});

}

}

public async Task RecordProjectHealthMetrics(string projectNumber, ProjectHealthMetrics health)

{

var tags = new[] { KeyValuePair.Create("project", projectNumber) };

\_metrics.RecordValue("project\_health\_score", health.OverallScore, tags);

\_metrics.RecordValue("coordination\_issues\_count", health.CoordinationIssues, tags);

\_metrics.RecordValue("files\_processed\_today", health.FilesProcessedToday, tags);

\_metrics.RecordValue("processing\_success\_rate", health.ProcessingSuccessRate, tags);

}

}

}nc Task ImplementUserInterfacesAsync()

{

*// Admin dashboard (React + TypeScript)*

await DeployApplicationAsync("admin-dashboard", new AdminDashboard());

*// Project health dashboard*

await DeployApplicationAsync("health-dashboard", new HealthDashboard());

*// API documentation (Swagger/OpenAPI)*

await DeployDocumentationAsync("api-docs", new APIDocumentation());

}

}

### **3. Disaster Recovery and Backup**

csharp

public class DisasterRecoveryService : IDisasterRecoveryService

{

public async Task<BackupStrategy> CreateBackupStrategyAsync()

{

return new BackupStrategy

{

*// Event Store - Point-in-time recovery*

EventStore = new BackupConfiguration

{

Type = BackupType.Continuous,

RetentionPolicy = TimeSpan.FromDays(90),

BackupFrequency = TimeSpan.FromHours(1),

CrossRegionReplication = true

},

*// Data Lake - Incremental backups*

DataLake = new BackupConfiguration

{

Type = BackupType.Incremental,

RetentionPolicy = TimeSpan.FromDays(365),

BackupFrequency = TimeSpan.FromDays(1),

VersioningEnabled = true

},

*// Read Models - Transaction log shipping*

ReadModels = new BackupConfiguration

{

Type = BackupType.TransactionLog,

RetentionPolicy = TimeSpan.FromDays(30),

BackupFrequency = TimeSpan.FromMinutes(15),

GeoRedundancy = true

},

*// Configuration and Secrets*

Configuration = new BackupConfiguration

{

Type = BackupType.Full,

RetentionPolicy = TimeSpan.FromDays(365),

BackupFrequency = TimeSpan.FromDays(1),

Encryption = true

}

};

}

public async Task<RecoveryPlan> CreateRecoveryPlanAsync()

{

return new RecoveryPlan

{

*// Recovery Time Objective: 4 hours*

RTO = TimeSpan.FromHours(4),

*// Recovery Point Objective: 15 minutes*

RPO = TimeSpan.FromMinutes(15),

RecoverySteps = new[]

{

new RecoveryStep

{

Order = 1,

Description = "Restore core infrastructure (Kubernetes, networking)",

EstimatedTime = TimeSpan.FromMinutes(30),

Prerequisites = new[] { "Infrastructure team notified", "Backup storage accessible" }

},

new RecoveryStep

{

Order = 2,

Description = "Restore event store from latest backup",

EstimatedTime = TimeSpan.FromMinutes(45),

Prerequisites = new

Retry

N

Continue

Edit

csharp

Prerequisites = new[] { "Infrastructure restored", "Event store cluster available" }

},

new RecoveryStep

{

Order = 3,

Description = "Restore data lake from incremental backups",

EstimatedTime = TimeSpan.FromHours(1),

Prerequisites = new[] { "Storage accounts accessible", "Network connectivity verified" }

},

new RecoveryStep

{

Order = 4,

Description = "Rebuild read models from event store",

EstimatedTime = TimeSpan.FromMinutes(90),

Prerequisites = new[] { "Event store operational", "PostgreSQL cluster restored" }

},

new RecoveryStep

{

Order = 5,

Description = "Deploy microservices and validate health",

EstimatedTime = TimeSpan.FromMinutes(45),

Prerequisites = new[] { "Container registry accessible", "Configuration restored" }

},

new RecoveryStep

{

Order = 6,

Description = "Resume file processing and validate end-to-end functionality",

EstimatedTime = TimeSpan.FromMinutes(30),

Prerequisites = new[] { "All services healthy", "Test files available" }

}

}

};

}

}

## **Performance Characteristics**

### **Expected Performance Metrics**

csharp

public class PerformanceExpectations

{

public static readonly PerformanceBenchmarks Benchmarks = new()

{

*// File Processing Throughput*

FileProcessing = new FileProcessingBenchmarks

{

SmallFiles = new ProcessingBenchmark

{

FileSize = "< 1MB",

ExpectedThroughput = "1000 files/hour per service instance",

AverageProcessingTime = TimeSpan.FromSeconds(3.6),

P95ProcessingTime = TimeSpan.FromSeconds(8)

},

MediumFiles = new ProcessingBenchmark

{

FileSize = "1MB - 50MB",

ExpectedThroughput = "200 files/hour per service instance",

AverageProcessingTime = TimeSpan.FromSeconds(18),

P95ProcessingTime = TimeSpan.FromSeconds(45)

},

LargeFiles = new ProcessingBenchmark

{

FileSize = "50MB - 500MB",

ExpectedThroughput = "20 files/hour per service instance",

AverageProcessingTime = TimeSpan.FromMinutes(3),

P95ProcessingTime = TimeSpan.FromMinutes(8)

},

ExtraLargeFiles = new ProcessingBenchmark

{

FileSize = "> 500MB",

ExpectedThroughput = "5 files/hour per service instance",

AverageProcessingTime = TimeSpan.FromMinutes(12),

P95ProcessingTime = TimeSpan.FromMinutes(30)

}

},

*// API Response Times*

APIPerformance = new APIPerformanceBenchmarks

{

SimpleQueries = new APIBenchmark

{

Description = "Get current project files",

ExpectedResponseTime = TimeSpan.FromMilliseconds(150),

P95ResponseTime = TimeSpan.FromMilliseconds(500)

},

ComplexQueries = new APIBenchmark

{

Description = "Coordination matrix with analytics",

ExpectedResponseTime = TimeSpan.FromMilliseconds(800),

P95ResponseTime = TimeSpan.FromSeconds(2)

},

SearchQueries = new APIBenchmark

{

Description = "Full-text search across all files",

ExpectedResponseTime = TimeSpan.FromMilliseconds(300),

P95ResponseTime = TimeSpan.FromSeconds(1)

}

},

*// Scalability Targets*

Scalability = new ScalabilityBenchmarks

{

MaxConcurrentFiles = 10\_000,

MaxProjectsSupported = 1\_000,

MaxFilesPerProject = 100\_000,

MaxConcurrentUsers = 500,

MaxAPIRequestsPerSecond = 1\_000,

MaxEventThroughput = 50\_000 *// events per second*

}

};

}

### **Resource Requirements**

yaml

*# Resource allocation guide*

production\_sizing:

small\_deployment: *# < 10 projects, < 10K files*

kubernetes\_nodes: 3

node\_size: "Standard\_D4s\_v3" *# 4 vCPU, 16GB RAM*

services:

file\_discovery: { replicas: 2, cpu: "500m", memory: "1Gi" }

classification: { replicas: 2, cpu: "1000m", memory: "2Gi" }

cad\_processor: { replicas: 2, cpu: "2000m", memory: "4Gi" }

pdf\_processor: { replicas: 2, cpu: "1000m", memory: "2Gi" }

data\_lake\_writer: { replicas: 2, cpu: "500m", memory: "1Gi" }

medium\_deployment: *# 10-100 projects, 10K-100K files*

kubernetes\_nodes: 6

node\_size: "Standard\_D8s\_v3" *# 8 vCPU, 32GB RAM*

services:

file\_discovery: { replicas: 3, cpu: "1000m", memory: "2Gi" }

classification: { replicas: 4, cpu: "1000m", memory: "2Gi" }

cad\_processor: { replicas: 6, cpu: "4000m", memory: "8Gi" }

pdf\_processor: { replicas: 4, cpu: "2000m", memory: "4Gi" }

ocr\_service: { replicas: 3, cpu: "2000m", memory: "4Gi" }

semantic\_analysis: { replicas: 2, cpu: "2000m", memory: "4Gi" }

data\_lake\_writer: { replicas: 3, cpu: "1000m", memory: "2Gi" }

large\_deployment: *# 100+ projects, 100K+ files*

kubernetes\_nodes: 12

node\_size: "Standard\_D16s\_v3" *# 16 vCPU, 64GB RAM*

specialized\_nodes:

cad\_processing: { count: 4, size: "Standard\_D8s\_v3", gpu: true }

ocr\_processing: { count: 2, size: "Standard\_NC6s\_v3", gpu: true }

services:

file\_discovery: { replicas: 5, cpu: "1000m", memory: "2Gi" }

classification: { replicas: 8, cpu: "1000m", memory: "2Gi" }

cad\_processor: { replicas: 12, cpu: "6000m", memory: "12Gi" }

pdf\_processor: { replicas: 8, cpu: "2000m", memory: "4Gi" }

ocr\_service: { replicas: 6, cpu: "4000m", memory: "8Gi" }

semantic\_analysis: { replicas: 4, cpu: "4000m", memory: "8Gi" }

data\_lake\_writer: { replicas: 6, cpu: "1000m", memory: "2Gi" }

## **Implementation Roadmap**

### **Phase 1: Foundation (Months 1-3)**

**Goal**: Basic file processing with core services

mermaid

gantt

title Phase 1: Foundation Implementation

dateFormat YYYY-MM-DD

section Infrastructure

Setup K8s Cluster :2024-01-01, 2w

Deploy Event Bus :2024-01-15, 1w

Setup Data Lake :2024-01-22, 2w

section Core Services

File Discovery Service :2024-02-01, 3w

Classification Service :2024-02-15, 3w

Basic Content Processing :2024-03-01, 4w

section APIs

Command API :2024-02-01, 2w

Query API :2024-02-15, 2w

Health Monitoring :2024-03-01, 1w

**Deliverables**:

* ✅ Event-driven infrastructure with Kafka/Event Hubs
* ✅ File discovery and classification services
* ✅ Basic metadata extraction (file properties, naming convention parsing)
* ✅ Data lake with Delta Lake format
* ✅ REST APIs for file operations
* ✅ Basic monitoring and health checks

### **Phase 2: Advanced Processing (Months 4-6)**

**Goal**: Specialized content extraction and analysis

mermaid

gantt

title Phase 2: Advanced Processing

dateFormat YYYY-MM-DD

section Content Processing

CAD Metadata Extraction :2024-04-01, 6w

PDF Processing with OCR :2024-04-15, 6w

BIM Model Processing :2024-05-01, 4w

section Analytics

Semantic Analysis :2024-05-01, 4w

Coordination Analysis :2024-05-15, 3w

Project Health Metrics :2024-06-01, 2w

section Optimization

Performance Tuning :2024-05-15, 4w

Caching Implementation :2024-06-01, 2w

**Deliverables**:

* ✅ CAD file processing with AutoCAD .NET API integration
* ✅ Advanced PDF processing with technical drawing OCR
* ✅ BIM model metadata extraction (Revit integration)
* ✅ Semantic analysis with NLP for technical content
* ✅ Real-time coordination analysis and conflict detection
* ✅ Project health monitoring dashboards
* ✅ Performance optimizations and intelligent caching

### **Phase 3: Enterprise Features (Months 7-9)**

**Goal**: Production-ready with enterprise integrations

mermaid

gantt

title Phase 3: Enterprise Features

dateFormat YYYY-MM-DD

section User Experience

Admin Dashboard :2024-07-01, 4w

Real-time Notifications :2024-07-15, 2w

Natural Language Query :2024-08-01, 3w

section Integrations

ERP Integration :2024-07-01, 4w

Webhook Management :2024-07-15, 2w

File Sync Services :2024-08-01, 3w

section Operations

Advanced Monitoring :2024-08-01, 3w

Security Hardening :2024-08-15, 2w

Disaster Recovery :2024-09-01, 2w

**Deliverables**:

* ✅ React-based admin dashboard with real-time updates
* ✅ Natural language query interface with AI
* ✅ ERP integrations (Deltek, Sage, Oracle)
* ✅ Advanced search with Elasticsearch
* ✅ Comprehensive monitoring and alerting
* ✅ Security hardening and compliance features
* ✅ Disaster recovery and backup procedures

### **Phase 4: Advanced Analytics (Months 10-12)**

**Goal**: Predictive analytics and AI-driven insights

**Deliverables**:

* 🔮 Predictive project health analytics
* 🔮 Automated quality assurance checks
* 🔮 Machine learning for content classification
* 🔮 Advanced reporting and business intelligence
* 🔮 Mobile applications for field access
* 🔮 Integration marketplace for third-party tools

## **Total Cost of Ownership (TCO)**

### **Infrastructure Costs (Annual)**

yaml

cloud\_infrastructure\_costs:

small\_deployment: *# < 10 projects*

azure\_kubernetes\_service: $15,000

data\_lake\_storage: $3,000

database\_services: $8,000

event\_streaming: $4,000

monitoring\_logging: $2,000

total\_annual: $32,000

medium\_deployment: *# 10-100 projects*

azure\_kubernetes\_service: $45,000

data\_lake\_storage: $12,000

database\_services: $24,000

event\_streaming: $12,000

monitoring\_logging: $6,000

total\_annual: $99,000

large\_deployment: *# 100+ projects*

azure\_kubernetes\_service: $120,000

data\_lake\_storage: $35,000

database\_services: $60,000

event\_streaming: $25,000

monitoring\_logging: $15,000

total\_annual: $255,000

development\_costs:

initial\_development: $500,000 *# 6-month team of 8 developers*

ongoing\_maintenance: $200,000 *# Annual maintenance and updates*

third\_party\_licenses: $50,000 *# AutoCAD API, OCR services, etc.*

roi\_projection:

manual\_processing\_cost\_savings: $300,000 *# Annual savings from automation*

reduced\_coordination\_errors: $150,000 *# Savings from early error detection*

faster\_project\_delivery: $200,000 *# Revenue from 10% faster delivery*

total\_annual\_benefits: $650,000

payback\_period: "18 months"

three\_year\_roi: "285%"

## **Key Success Metrics**

### **Technical Metrics**

* **Processing Throughput**: 95th percentile under 30 seconds for files < 50MB
* **System Availability**: 99.9% uptime (8.77 hours downtime per year)
* **API Response Time**: 95th percentile under 500ms for simple queries
* **Data Consistency**: 99.99% accuracy in file classification and metadata extraction
* **Scalability**: Linear scaling to 10,000 concurrent file processing operations

### **Business Metrics**

* **Time to Insight**: Reduce project status assessment time by 80%
* **Error Reduction**: 70% reduction in coordination errors through automated detection
* **Productivity Gain**: 50% reduction in manual file management tasks
* **Project Delivery**: 15% improvement in on-time project delivery
* **User Adoption**: 90% of engineering staff actively using the system within 6 months

## **Why This Architecture is Optimal**

### **1. Maximum Scalability**

* **Horizontal Scaling**: Each service scales independently based on demand
* **Resource Optimization**: Different services can run on appropriately sized hardware
* **Global Distribution**: Can be deployed across multiple regions for worldwide firms
* **Auto-scaling**: Kubernetes HPA and custom metrics ensure optimal resource utilization

### **2. Ultimate Ease of Use**

* **Natural Language Interface**: Engineers can query using plain English
* **Real-time Dashboards**: Instant visibility into project health and file status
* **Automated Processing**: Zero-touch file processing with intelligent classification
* **Rich APIs**: Easy integration with existing tools and workflows

### **3. Production Robustness**

* **Event Sourcing**: Complete audit trail and ability to replay/recover from any point
* **CQRS**: Optimized read and write patterns for different use cases
* **Circuit Breakers**: Fault isolation prevents cascading failures
* **Multi-tier Caching**: Sub-second response times for common queries

### **4. Future-Proof Design**

* **Microservices**: Easy to add new file types, processing capabilities, or integrations
* **Event-Driven**: New services can subscribe to existing events without system changes
* **Cloud-Native**: Can run on any Kubernetes platform (Azure, AWS, GCP, on-premises)
* **AI-Ready**: Architecture supports ML model integration for advanced analytics

This architecture represents the pinnacle of modern software design applied to the unique challenges of AEC file processing. It combines the scalability of cloud-native microservices, the reliability of event sourcing, the performance of CQRS, and the user experience of modern web applications to create a system that will serve engineering firms for years to come.

The investment in this architecture pays dividends through reduced manual work, faster project delivery, improved quality, and the foundation for advanced analytics and AI-driven insights that will become increasingly important in the AEC industry.