Technical Architecture: Process Manufacturing Enhancements Suite (D365-PMA)

Executive Architecture Summary

Solution: Dynamics 365 Process Manufacturing Accelerator (D365-PMA)
Architecture Pattern: Native Business Central extension with cloud-enhanced capabilities
Primary Integration: Dynamics 365 Business Central AL Framework + Power Platform
Target Scale: 100+ process manufacturing sites within 2 years
Performance: Multi-output batch processing with real-time recipe management

System Architecture Overview

High-Level Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                    Microsoft Cloud Platform                     │
├─────────────────────────────────────────────────────────────────┤
│  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐ │
│  │ Dynamics 365    │  │ Power Platform  │  │   Azure AI      │ │
│  │ Business        │◄─┤ Apps & BI       ├─►│   Services      │ │
│  │ Central         │  └─────────────────┘  └─────────────────┘ │
│  └─────────────────┘           │                     │         │
│           │           ┌─────────────────┐  ┌─────────────────┐  │
│           │           │   Dataverse     │  │ Azure Machine   │  │
│           └──────────►│   Storage       │◄─┤ Learning        │  │
│                       └─────────────────┘  └─────────────────┘  │
└─────────────────────────────────────────────────────────────────┘
                               │
                    ┌─────────────────┐
                    │  API Gateway    │
                    │  Management     │
                    └─────────────────┘
                               │
┌─────────────────────────────────────────────────────────────────┐
│                Process Manufacturing Systems                     │
├─────────────────────────────────────────────────────────────────┤
│  ┌─────────────┐  ┌─────────────┐  ┌─────────────┐  ┌─────────┐ │
│  │    LIMS     │  │    DCS      │  │Process      │  │Quality  │ │
│  │   Systems   │  │   SCADA     │  │Historians   │  │Systems  │ │
│  └─────────────┘  └─────────────┘  └─────────────┘  └─────────┘ │
└─────────────────────────────────────────────────────────────────┘
```

Core Components Architecture

1. Business Central Extensions (AL Framework)

Recipe Management Extension

- **Technology**: AL Language for Business Central
- **Object Types**: Tables, Pages, Codeunits, Reports
- **Core Tables**:
  - Recipe Header (Recipe ID, Version, Status, Approval)
  - Recipe Lines (Ingredients, Quantities, Process Steps)
  - Formula Versions (Change tracking, Approval workflow)
  - Process Parameters (Temperature, Pressure, Time, pH)

Multi-Output Production Extension

- **Technology**: Enhanced Manufacturing module tables
- **New Objects**:
  - Co-Product Setup (Item relationships, yield factors)
  - By-Product Configuration (Automatic creation rules)
  - Joint Cost Allocation (Cost distribution methods)
  - Yield Tracking (Actual vs. theoretical analysis)

Process Control Integration

- **Technology**: Web Services and REST APIs
- **Capabilities**:
  - Real-time parameter monitoring
  - Statistical Process Control (SPC) calculations
  - Control limit management and alerting
  - Process deviation recording and analysis

2. Recipe Management System

Data Model Design

```
table 50001 "PMA Recipe Header"
{
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Recipe No."; Code[20])
        {
            Caption = 'Recipe No.';
        }
        field(2; "Version No."; Code[10])
        {
            Caption = 'Version No.';
        }
        field(3; "Item No."; Code[20])
        {
            Caption = 'Item No.';
            TableRelation = Item;
        }
        field(4; "Description"; Text[100])
        {
            Caption = 'Description';
        }
        field(5; "Status"; Enum "PMA Recipe Status")
        {
            Caption = 'Status';
        }
        field(6; "Batch Size"; Decimal)
        {
            Caption = 'Batch Size';
            DecimalPlaces = 0:5;
        }
        field(7; "Unit of Measure"; Code[10])
        {
            Caption = 'Unit of Measure';
            TableRelation = "Unit of Measure";
        }
        field(8; "Expected Yield %"; Decimal)
        {
            Caption = 'Expected Yield %';
            DecimalPlaces = 2:5;
        }
        field(9; "Approved By"; Code[50])
        {
            Caption = 'Approved By';
            TableRelation = User."User Name";
        }
        field(10; "Approved Date"; DateTime)
        {
            Caption = 'Approved Date';
        }
    }
}

table 50002 "PMA Recipe Line"
{
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Recipe No."; Code[20])
        {
            Caption = 'Recipe No.';
            TableRelation = "PMA Recipe Header"."Recipe No.";
        }
        field(2; "Version No."; Code[10])
        {
            Caption = 'Version No.';
        }
        field(3; "Line No."; Integer)
        {
            Caption = 'Line No.';
        }
        field(4; "Type"; Enum "PMA Recipe Line Type")
        {
            Caption = 'Type';
        }
        field(5; "No."; Code[20])
        {
            Caption = 'No.';
            TableRelation = IF (Type = CONST(Item)) Item
                           ELSE IF (Type = CONST(Resource)) Resource;
        }
        field(6; "Description"; Text[100])
        {
            Caption = 'Description';
        }
        field(7; "Quantity"; Decimal)
        {
            Caption = 'Quantity';
            DecimalPlaces = 0:5;
        }
        field(8; "Unit of Measure Code"; Code[10])
        {
            Caption = 'Unit of Measure Code';
        }
        field(9; "Process Step"; Integer)
        {
            Caption = 'Process Step';
        }
        field(10; "Critical Parameter"; Boolean)
        {
            Caption = 'Critical Parameter';
        }
    }
}
```

Recipe Scaling Algorithm

```
procedure ScaleRecipe(RecipeNo: Code[20]; VersionNo: Code[10]; NewBatchSize:
Decimal): Boolean
var
    RecipeHeader: Record "PMA Recipe Header";
    RecipeLine: Record "PMA Recipe Line";
    ScaleFactor: Decimal;
begin
    if RecipeHeader.Get(RecipeNo, VersionNo) then begin
        ScaleFactor := NewBatchSize / RecipeHeader."Batch Size";

        RecipeLine.SetRange("Recipe No.", RecipeNo);
        RecipeLine.SetRange("Version No.", VersionNo);
        if RecipeLine.FindSet() then
            repeat
                RecipeLine.Quantity := RecipeLine.Quantity * ScaleFactor;
                RecipeLine.Modify();
            until RecipeLine.Next() = 0;

        RecipeHeader."Batch Size" := NewBatchSize;
        RecipeHeader.Modify();
        exit(true);
    end;
    exit(false);
end;
```

3. Multi-Output Production System

Enhanced Production Order Structure

```
table 50010 "PMA Production Output"
{
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Production Order No."; Code[20])
        {
            Caption = 'Production Order No.';
            TableRelation = "Production Order"."No.";
        }
        field(2; "Line No."; Integer)
        {
            Caption = 'Line No.';
        }
        field(3; "Output Type"; Enum "PMA Output Type")
        {
            Caption = 'Output Type';
            // Main Product, Co-Product, By-Product
        }
        field(4; "Item No."; Code[20])
        {
            Caption = 'Item No.';
            TableRelation = Item;
        }
        field(5; "Expected Quantity"; Decimal)
        {
            Caption = 'Expected Quantity';
            DecimalPlaces = 0:5;
        }
        field(6; "Actual Quantity"; Decimal)
        {
            Caption = 'Actual Quantity';
            DecimalPlaces = 0:5;
        }
        field(7; "Yield %"; Decimal)
        {
            Caption = 'Yield %';
            DecimalPlaces = 2:5;
        }
        field(8; "Cost Allocation %"; Decimal)
        {
            Caption = 'Cost Allocation %';
            DecimalPlaces = 2:5;
        }
        field(9; "Unit Cost"; Decimal)
        {
            Caption = 'Unit Cost';
            DecimalPlaces = 2:5;
        }
        field(10; "Total Cost"; Decimal)
        {
            Caption = 'Total Cost';
            DecimalPlaces = 2:5;
        }
    }
}
```

Joint Cost Allocation Methods

```
enum 50001 "PMA Cost Allocation Method"
{
    Extensible = true;

    value(0; "Net Realizable Value")
    {
        Caption = 'Net Realizable Value';
    }
    value(1; "Physical Units")
    {
        Caption = 'Physical Units';
    }
    value(2; "Market Value")
    {
        Caption = 'Market Value';
    }
    value(3; "Standard Cost")
    {
        Caption = 'Standard Cost';
    }
}

procedure AllocateJointCosts(ProductionOrderNo: Code[20]; Method: Enum "PMA
Cost Allocation Method")
```

```
var
    ProductionOutput: Record "PMA Production Output";
    TotalCost: Decimal;
    AllocationBase: Decimal;
    TotalBase: Decimal;
begin
    // Calculate total production cost
    TotalCost := CalculateTotalProductionCost(ProductionOrderNo);

    // Calculate allocation base based on method
    ProductionOutput.SetRange("Production Order No.", ProductionOrderNo);
    if ProductionOutput.FindSet() then
        repeat
            case Method of
                Method::"Physical Units":
                    AllocationBase := ProductionOutput."Actual Quantity";
                Method::"Market Value":
                    AllocationBase := ProductionOutput."Actual Quantity" *
GetMarketPrice(ProductionOutput."Item No.");
                Method::"Net Realizable Value":
                    AllocationBase := CalculateNRV(ProductionOutput."Item No.",
ProductionOutput."Actual Quantity");
            end;
            TotalBase += AllocationBase;
        until ProductionOutput.Next() = 0;

    // Apply cost allocation
    if ProductionOutput.FindSet() then
        repeat
            ProductionOutput."Cost Allocation %" := (AllocationBase /
TotalBase) * 100;
            ProductionOutput."Total Cost" := TotalCost *
(ProductionOutput."Cost Allocation %" / 100);
            ProductionOutput."Unit Cost" := ProductionOutput."Total Cost" /
ProductionOutput."Actual Quantity";
            ProductionOutput.Modify();
        until ProductionOutput.Next() = 0;
end;
```

4. Process Control Integration

Statistical Process Control (SPC) Engine

```
table 50020 "PMA Process Parameter"
{
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Parameter Code"; Code[20])
        {
            Caption = 'Parameter Code';
        }
        field(2; "Description"; Text[100])
        {
            Caption = 'Description';
        }
        field(3; "Unit of Measure"; Code[10])
        {
            Caption = 'Unit of Measure';
        }
        field(4; "Target Value"; Decimal)
        {
            Caption = 'Target Value';
            DecimalPlaces = 0:5;
        }
        field(5; "Upper Control Limit"; Decimal)
        {
            Caption = 'Upper Control Limit';
            DecimalPlaces = 0:5;
        }
        field(6; "Lower Control Limit"; Decimal)
        {
            Caption = 'Lower Control Limit';
            DecimalPlaces = 0:5;
        }
        field(7; "Upper Spec Limit"; Decimal)
        {
            Caption = 'Upper Spec Limit';
            DecimalPlaces = 0:5;
        }
        field(8; "Lower Spec Limit"; Decimal)
        {
            Caption = 'Lower Spec Limit';
            DecimalPlaces = 0:5;
        }
        field(9; "Critical Parameter"; Boolean)
        {
            Caption = 'Critical Parameter';
        }
    }
}

procedure CalculateControlLimits(ParameterCode: Code[20]; SampleSize: Integer):
Boolean
var
    ProcessParameter: Record "PMA Process Parameter";
    ProcessData: Record "PMA Process Data";
    Mean: Decimal;
    StdDev: Decimal;
    A2Factor: Decimal;
begin
    if ProcessParameter.Get(ParameterCode) then begin
```

```
        // Calculate statistical parameters from historical data
        CalculateStatistics(ParameterCode, Mean, StdDev);

        // Get A2 factor based on sample size (SPC tables)
        A2Factor := GetA2Factor(SampleSize);

        // Calculate control limits (3-sigma)
        ProcessParameter."Upper Control Limit" := Mean + (A2Factor * StdDev);
        ProcessParameter."Lower Control Limit" := Mean - (A2Factor * StdDev);
        ProcessParameter."Target Value" := Mean;
        ProcessParameter.Modify();

        exit(true);
    end;
    exit(false);
end;
```

Integration Architecture

Power Platform Integration

Power Apps Mobile Application

- **Target Users**: Production operators, quality inspectors
- **Key Features**:
  - Batch record creation and management
  - Real-time parameter entry and validation
  - Photo capture for quality documentation
  - Offline capability with sync when connected

Power BI Analytics Dashboard

- **Embedded Analytics**: Native BC integration with Power BI
- **Key Reports**:
  - Yield trend analysis by recipe and time period
  - Process parameter control charts
  - Cost allocation analysis across products
  - Regulatory compliance reporting

Power Automate Workflows

- **Recipe Approval**: Automated workflow with email notifications
- **Exception Handling**: Alerts for process deviations
- **Quality Notifications**: Automatic quality order creation
- **Regulatory Reporting**: Scheduled report generation and distribution

External System Integration

LIMS (Laboratory Information Management System)

- **Integration Method**: REST APIs with OAuth 2.0 authentication
- **Data Exchange**:
  - Sample creation and tracking
  - Test result import and validation
  - Certificate of Analysis (CoA) generation
  - Quality specification management

DCS/SCADA Integration

- **Protocols**: OPC-UA, Modbus TCP, Ethernet/IP
- **Data Types**:
  - Real-time process parameters
  - Batch event notifications
  - Equipment status and alarms
  - Historical trend data

ERP Integration (Multi-Company)

- **Method**: Web services and data export/import
- **Synchronization**:
  - Master data (items, customers, vendors)
  - Production orders and schedules
  - Inventory transactions
  - Cost accounting data

Regulatory Compliance Framework

FDA 21 CFR Part 11 Compliance

Electronic Records Management

```
table 50030 "PMA Electronic Record"
{
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Record ID"; Code[20])
        {
            Caption = 'Record ID';
        }
        field(2; "Record Type"; Enum "PMA Record Type")
        {
            Caption = 'Record Type';
        }
        field(3; "Batch No."; Code[20])
        {
            Caption = 'Batch No.';
        }
        field(4; "Created By"; Code[50])
        {
            Caption = 'Created By';
        }
        field(5; "Created DateTime"; DateTime)
        {
            Caption = 'Created DateTime';
        }
        field(6; "Modified By"; Code[50])
        {
            Caption = 'Modified By';
        }
        field(7; "Modified DateTime"; DateTime)
        {
            Caption = 'Modified DateTime';
        }
        field(8; "Digital Signature"; Text[250])
        {
            Caption = 'Digital Signature';
        }
        field(9; "Audit Trail"; Blob)
        {
            Caption = 'Audit Trail';
        }
        field(10; "Locked"; Boolean)
        {
            Caption = 'Locked';
        }
    }
}
```

Digital Signature Implementation

```
procedure ApplyDigitalSignature(RecordID: Code[20]; UserID: Code[50]): Boolean
var
    ElectronicRecord: Record "PMA Electronic Record";
    UserSetup: Record "User Setup";
    DigitalCert: Record "PMA Digital Certificate";
    SignatureHash: Text[250];
begin
    if ElectronicRecord.Get(RecordID) then begin
        // Verify user has valid digital certificate
        if DigitalCert.Get(UserID) and (DigitalCert."Expiry Date" > Today) then
begin
            // Generate signature hash
            SignatureHash := GenerateSignatureHash(RecordID, UserID,
CurrentDateTime);

            // Apply signature and lock record
            ElectronicRecord."Digital Signature" := SignatureHash;
            ElectronicRecord."Locked" := true;
            ElectronicRecord.Modify();

            // Log audit trail
            LogAuditEvent(RecordID, 'Digital Signature Applied', UserID);
            exit(true);
        end;
    end;
    exit(false);
end;
```

Batch Genealogy and Traceability

Complete Lot Tracking System

```
table 50040 "PMA Batch Genealogy"
{
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Batch No."; Code[20])
        {
            Caption = 'Batch No.';
        }
        field(2; "Parent Batch No."; Code[20])
        {
            Caption = 'Parent Batch No.';
        }
        field(3; "Item No."; Code[20])
        {
            Caption = 'Item No.';
        }
        field(4; "Lot No."; Code[50])
        {
            Caption = 'Lot No.';
        }
        field(5; "Production Date"; Date)
        {
            Caption = 'Production Date';
        }
        field(6; "Expiry Date"; Date)
        {
            Caption = 'Expiry Date';
        }
        field(7; "Supplier Lot No."; Code[50])
        {
            Caption = 'Supplier Lot No.';
        }
        field(8; "Quality Status"; Enum "PMA Quality Status")
        {
            Caption = 'Quality Status';
        }
        field(9; "Customer Shipments"; Integer)
        {
            Caption = 'Customer Shipments';
        }
        field(10; "Recall Status"; Enum "PMA Recall Status")
        {
            Caption = 'Recall Status';
        }
    }
}

procedure TraceForward(BatchNo: Code[20]): List of [Code[20]]
var
    BatchGenealogy: Record "PMA Batch Genealogy";
    CustomerLedger: Record "Cust. Ledger Entry";
    ShipmentLines: Record "Sales Shipment Line";
    CustomerList: List of [Code[20]];
begin
    // Find all downstream customers who received this batch
    BatchGenealogy.SetRange("Parent Batch No.", BatchNo);
    if BatchGenealogy.FindSet() then
        repeat
            ShipmentLines.SetRange("Lot No.", BatchGenealogy."Lot No.");
            if ShipmentLines.FindSet() then
                repeat
                    if not CustomerList.Contains(ShipmentLines."Sell-to
Customer No.") then
                        CustomerList.Add(ShipmentLines."Sell-to Customer No.");
                until ShipmentLines.Next() = 0;
        until BatchGenealogy.Next() = 0;

    exit(CustomerList);
end;
```

Performance and Scalability

Business Central Extension Performance

Object Design Optimization

- **Table Design**: Proper indexing on frequently queried fields
- **Page Performance**: Minimize OnAfterGetRecord triggers
- **Report Optimization**: Use temporary tables for complex calculations
- **Web Service Efficiency**: Pagination for large data sets

Multi-Company Support

```
procedure SynchronizeRecipeAcrossCompanies(RecipeNo: Code[20]; VersionNo:
Code[10])
var
    Company: Record Company;
    RecipeHeader: Record "PMA Recipe Header";
    RecipeLine: Record "PMA Recipe Line";
begin
    if Company.FindSet() then
        repeat
            RecipeHeader.ChangeCompany(Company.Name);
            if not RecipeHeader.Get(RecipeNo, VersionNo) then begin
                // Copy recipe to target company
                CopyRecipeToCompany(RecipeNo, VersionNo, Company.Name);
            end;
        until Company.Next() = 0;
end;
```

Scalability Architecture

Multi-Site Deployment

- **Database**: Separate BC environments per region/site
- **Master Data**: Centralized synchronization via web services
- **Reporting**: Consolidated analytics using Power BI
- **Performance**: Local processing with cloud aggregation

Performance Benchmarks

- **Recipe Processing**: 1000+ active recipes per site
- **Batch Operations**: 500+ concurrent batch processes
- **Data Volume**: 10GB+ historical process data per site
- **User Capacity**: 200+ concurrent users per site

Implementation Plan

Phase 1: Core Extension Development (Months 1-6)

Development Team Requirements

- **BC Architect**: 1 FTE (AL development, manufacturing expertise)
- **AL Developers**: 2 FTE (Business Central extensions)
- **Power Platform Developer**: 1 FTE (Power Apps, Power BI)
- **QA Engineer**: 1 FTE (Testing, validation)
- **Regulatory Consultant**: 0.5 FTE (FDA/EMA compliance)

Key Deliverables

- Recipe management system with version control
- Multi-output production order processing

- Basic yield tracking and cost allocation
- Power Apps mobile application for batch recording
- Integration with standard BC manufacturing modules

Success Criteria

- 3 pilot customers processing multi-output batches
- Recipe scaling and costing functionality validated
- Regulatory compliance framework implemented
- Customer feedback incorporation and iteration

Phase 2: Advanced Features (Months 7-12)

Additional Team Requirements

- **Integration Specialist**: 1 FTE (LIMS, DCS integration)
- **Process Engineer**: 0.5 FTE (SPC, quality systems)

Key Deliverables

- Statistical Process Control (SPC) capabilities
- LIMS integration for quality data
- Advanced analytics and reporting
- Regulatory compliance documentation
- Multi-site deployment capabilities

Success Criteria

- 15+ customer deployments across chemical and food industries
- Regulatory audit success at pilot customers
- Advanced analytics demonstrating process optimization
- Partner channel establishment for implementation

Phase 3: Market Expansion (Months 13-18)

Key Deliverables

- Industry-specific templates (pharma, specialty chemicals)
- Global localization and regulatory compliance
- Advanced AI/ML for process optimization
- Comprehensive training and certification programs
- Microsoft AppSource marketplace listing

Success Criteria

- 50+ customer deployments with documented ROI
- International market expansion (EU, Asia-Pacific)
- Industry recognition and analyst coverage
- Sustainable revenue growth with customer success metrics

Risk Assessment and Mitigation

Technical Risks

Business Central Platform Limitations (Medium Impact, Low Probability)

- **Risk**: AL framework constraints limiting process manufacturing functionality
- **Mitigation**: Early Microsoft engagement, alternative Power Platform approaches
- **Timeline Impact**: +3 months for alternative implementation

Integration Complexity with Legacy Systems (High Impact, Medium Probability)

- **Risk**: LIMS and DCS integration challenges affecting adoption
- **Mitigation**: Standard API development, partner ecosystem for implementation
- **Investment**: +$300K for integration framework development

Regulatory Compliance Evolution (Medium Impact, Medium Probability)

- **Risk**: Changing FDA/EMA requirements affecting compliance features
- **Mitigation**: Active regulatory monitoring, compliance consultant engagement
- **Resource**: Ongoing regulatory consultant retainer

Business Risks

Market Adoption in Conservative Industries (Low Impact, Medium Probability)

- **Risk**: Process manufacturers slow to adopt new BC-based solutions
- **Mitigation**: Strong industry references, compliance demonstration
- **Strategy**: Extended pilot programs, industry association participation

Competition from Specialized Process ERP (Medium Impact, Medium Probability)

- **Risk**: Established process manufacturing vendors enhancing BC integration
- **Mitigation**: Superior BC integration, Microsoft partnership advantages
- **Investment**: Continuous R&D and feature development

Customer Implementation Complexity (Medium Impact, High Probability)

- **Risk**: Complex implementations affecting customer satisfaction
- **Mitigation**: Comprehensive training, certified partner network
- **Resource**: Customer success team and implementation methodology

Resource Requirements Summary

Development Investment (18 months)

- **Total FTEs**: 8-10 team members across phases
- **Estimated Cost**: $2.1M in development resources
- **Specialized Skills**: BC AL development, process manufacturing, regulatory compliance

Infrastructure and Tools

- **Development Environment**: BC on-premises + cloud development
- **Testing Infrastructure**: $75K for multi-environment testing
- **Regulatory Compliance**: $150K for validation and documentation

Go-to-Market Investment

- **Microsoft Partnership**: BC ISV program participation
- **Industry Marketing**: $300K for trade shows, content marketing
- **Partner Channel**: $200K for partner training and enablement

Total Initial Investment: $2.8M over 18 months
Break-even: Month 15 with 30+ customer deployments
ROI: 280% by month 30 with target market penetration

Conclusion

The Process Manufacturing Enhancements Suite provides a comprehensive technical solution that transforms Dynamics 365 Business Central into a full-featured process manufacturing platform. The native AL extension approach ensures seamless integration while the Power Platform components provide modern user experiences and advanced analytics.

The regulatory compliance framework and industry-specific features position this solution to capture significant market share in the chemical, food, and pharmaceutical industries seeking to modernize their operations with Microsoft technology while meeting strict regulatory requirements.