18 Multi Architecture Perspectives

2025-10-19

Contents

1 Multi-Architecture Perspectives

1.1 Enterprise, Data, Integration & Business Architecture

Document Version: 1.0 **Last Updated:** 2025-10-18 **Status:** Complete

---

1.2 Table of Contents

1.3 1. Introduction

1.3.1 1.1 Purpose

This document provides comprehensive architectural views from four critical perspectives: - **Enterprise Architecture:** Strategic technology landscape, standards, principles - **Data Architecture:** Data models, flows, governance, quality - **Integration Architecture:** System interfaces, APIs, message patterns - **Business Architecture:** Value streams, capabilities, processes

1.3.2 1.2 Architecture Framework

We use **TOGAF 9.2** (The Open Group Architecture Framework) as our foundation, adapted for robotics and manufacturing domains.

```
                    TOGAF Architecture Domains


    Business              Data
    Architecture          Architecture
```

- Value streams
- Capabilities
- Processes
- Data models
- Data flows
- Governance

Application Architecture

- Application portfolio
- Integration patterns
- APIs & interfaces

Technology Architecture

- Infrastructure
- Platforms & tools
- Standards

1.3.3 1.3 Stakeholders

Role	Concerns	Primary Artifacts
Enterprise Architect	Strategic alignment, standards, portfolio	EA roadmap, principles, standards catalog
Data Architect	Data quality, governance, integration	Data models, lineage, quality rules
Integration Architect	Connectivity, APIs, message flows	Integration patterns, API specs, ESB design
Business Architect	Value delivery, capabilities, processes	Value stream maps, capability models

---

1.4 2. Enterprise Architecture

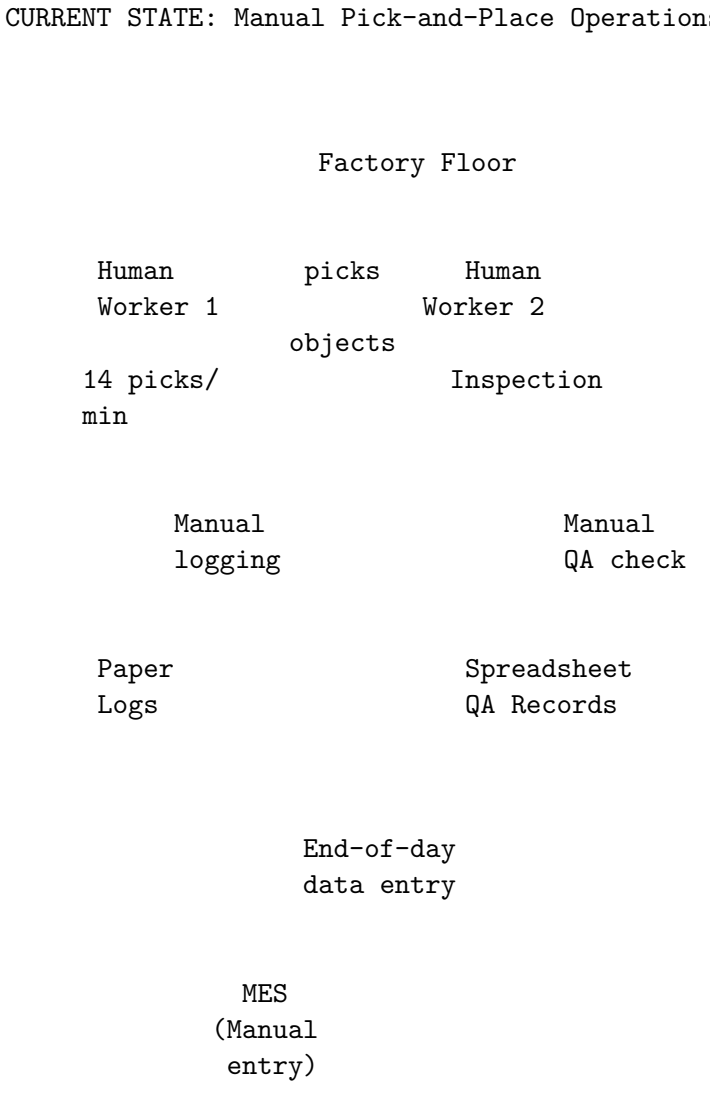
1.4.1 2.1 Enterprise Architecture Vision

Mission: Establish a scalable, secure, and maintainable robotics automation platform that enables digital transformation in manufacturing operations.

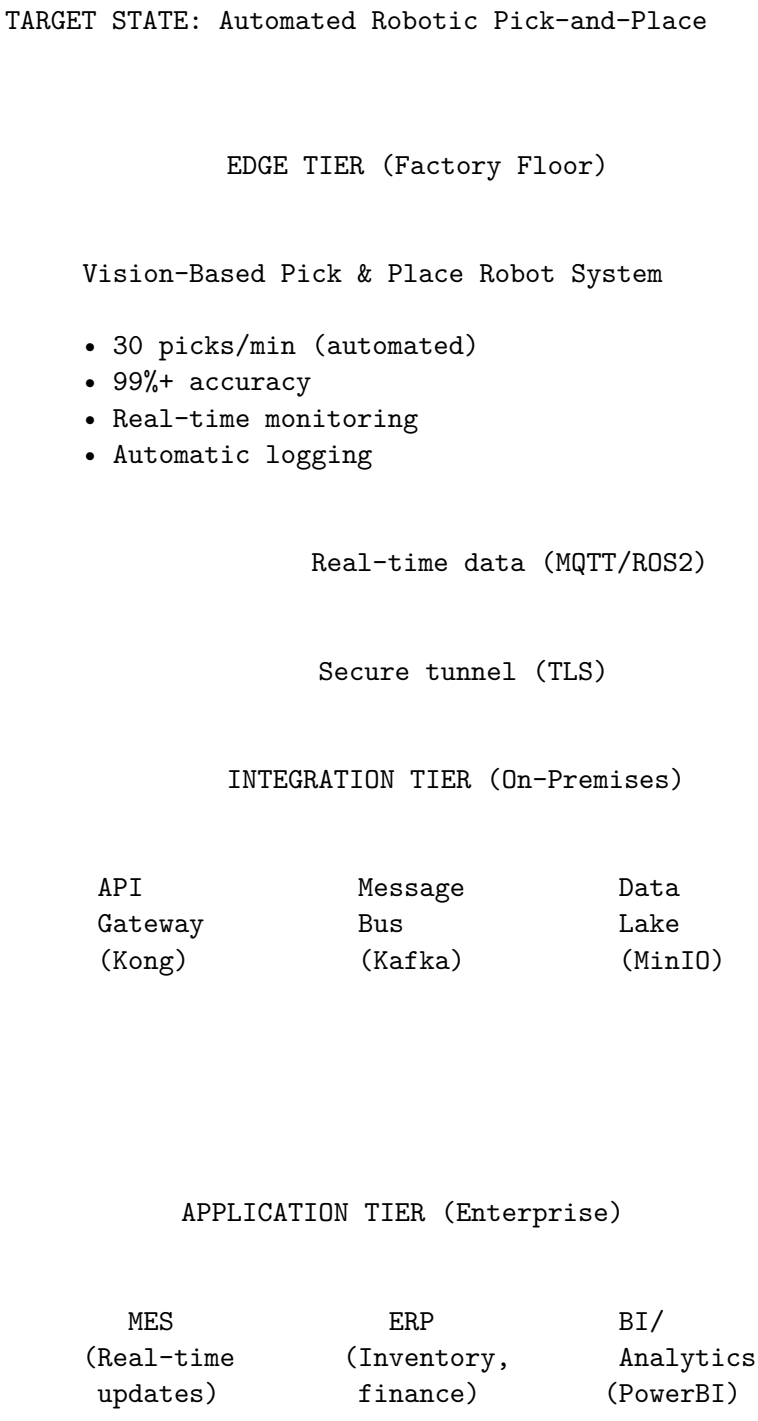
Strategic Objectives: 1. **Standardization:** Common platforms, tools, and patterns across all robotic systems 2. **Scalability:** Support 1-100 robot deployments with minimal incremental effort

3. **Interoperability:** Seamless integration with existing enterprise systems (MES, ERP, SCADA) 4. **Innovation:** Enable rapid adoption of new AI/ML models and technologies 5. **Security:** Enterprise-grade security, compliance, and audit capabilities

1.4.2 2.2 Current State Architecture (As-Is)



1.4.3 2.3 Target State Architecture (To-Be)



```
Automated data integration (zero manual entry)
Labor savings ($175k/year)
Scalable to 100+ robots
```

1.4.4 2.4 Technology Standards & Principles

1.4.4.1 2.4.1 Architecture Principles

Principle	Statement	Rationale	Implications
P1: Cloud-Native	Use containerized, microservices architecture	Scalability, portability, rapid deployment	All services run in Docker/K8s
P2: API-First	All integrations via well-defined APIs	Loose coupling, reusability	REST/gRPC APIs for all services
P3: Data-Driven	Collect, store, and analyze all operational data	Continuous improvement, predictive maintenance	Centralized data lake, ML pipelines
P4: Security by Design	Security integrated from the start	Compliance, risk mitigation	Zero-trust, encryption, audit logs
P5: Open Standards	Prefer open-source and industry standards	Vendor independence, community support	ROS2, MQTT, OPC-UA, REST
P6: Modularity	Components are independently deployable	Flexibility, maintainability	Microservices, pluggable modules
P7: Automation	Automate deployment, testing, monitoring	Speed, consistency, reliability	CI/CD, IaC, automated testing

1.4.4.2 2.4.2 Technology Standards Catalog

Category	Standard	Rationale	Alternatives Considered
Robotics Middleware	ROS2 Humble	Industry standard, real-time support, large ecosystem	ROS1 (legacy), proprietary SDKs
Containerization	Docker 24.x	Portability, reproducibility, ecosystem	Podman, LXC
Orchestration	Kubernetes 1.28	Scalability, high availability, declarative	Docker Swarm, Nomad
API Gateway	Kong 3.x	Performance, plugins, open-source	Nginx, Traefik, AWS API Gateway

Category	Standard	Rationale	Alternatives Considered
Message Broker	Apache Kafka 3.5	High throughput, fault tolerance, replay capability	RabbitMQ, MQTT Broker, Redis Streams
Time-Series DB	InfluxDB 2.7	Optimized for metrics, retention policies	Prometheus, TimescaleDB
Relational DB	PostgreSQL 15	ACID, mature, JSON support	MySQL, SQL Server
Object Storage	MinIO (S3-compatible)	Scalable, S3 API, on-premises	AWS S3, Azure Blob, Ceph
Monitoring	Prometheus + Grafana	De facto standard, rich ecosystem	Datadog, New Relic, Elastic APM
Logging	ELK Stack (Elasticsearch, Logstash, Kibana)	Centralized, searchable, scalable	Splunk, Graylog, Loki
CI/CD	GitHub Actions	Native GitHub integration, free for open-source	Jenkins, GitLab CI, CircleCI
IaC	Terraform + Ansible	Multi-cloud, declarative, mature	CloudFormation, Pulumi

1.4.5 2.5 Application Portfolio

```
APPLICATION PORTFOLIO MAP


STRATEGIC IMPORTANCE
(High)


INVEST                    MAINTAIN

• Vision AI Platform      • ROS2 Control System
• MLOps Pipeline          • MES Integration
• Digital Twin            • ERP Connector


ELIMINATE                 TOLERATE

• Paper Logs (legacy)     • Manual Calibration
• Spreadsheet QA            (migrate to auto)
```

```
                    (Low)
                BUSINESS VALUE
                ←            →
```

1.4.6 2.6 EA Roadmap (3-Year)

```
YEAR 1 (2025): Foundation

Q1-Q2: Single robot system deployment (MVP)
Q3:    MES/ERP integration
Q4:    Monitoring & analytics platform

YEAR 2 (2026): Scale

Q1:    5-robot deployment (multi-cell)
Q2:    Digital twin & simulation
Q3:    Predictive maintenance (ML)
Q4:    Edge computing optimization

YEAR 3 (2027): Transform

Q1:    20+ robot fleet management
Q2:    Cross-facility orchestration
Q3:    Advanced AI (reinforcement learning)
Q4:    Zero-downtime autonomous operation
```

1.5 3. Data Architecture

1.5.1 3.1 Data Architecture Overview

Mission: Establish a unified, governed, and high-quality data foundation to support real-time operations, analytics, and AI/ML.

1.5.2 3.2 Conceptual Data Model

```
            CORE ENTITIES & RELATIONSHIPS


              Robot
```

- robot_id
- model
- location

1
operates
n

Pick Task

- task_id
- timestamp
- status

1
picks
n

Object

- object_id
- class_name
- pose

n
uses
1

Grasp

- grasp_id
- pose
- quality

1.5.3 3.3 Logical Data Model (Relational)

```sql
-- ROBOTS
CREATE TABLE robots (
    robot_id VARCHAR(50) PRIMARY KEY,
    model VARCHAR(100) NOT NULL,
    serial_number VARCHAR(100) UNIQUE NOT NULL,
    location VARCHAR(100),
    status VARCHAR(20) DEFAULT 'offline',  -- offline, idle, running, error
    last_heartbeat TIMESTAMP,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

-- PICKS (Transactional data)
CREATE TABLE picks (
```

```sql
    pick_id SERIAL PRIMARY KEY,
    robot_id VARCHAR(50) REFERENCES robots(robot_id),
    timestamp TIMESTAMP NOT NULL DEFAULT NOW(),

    -- Object details
    object_class VARCHAR(50),
    object_pose JSONB,  -- {x, y, z, qx, qy, qz, qw}

    -- Grasp details
    grasp_pose JSONB,
    grasp_quality FLOAT CHECK (grasp_quality BETWEEN 0 AND 1),
    grasp_force FLOAT,  -- Newtons

    -- Place details
    place_pose JSONB,
    place_accuracy FLOAT,  -- mm

    -- Performance metrics
    cycle_time FLOAT,  -- seconds
    vision_latency FLOAT,  -- ms
    planning_time FLOAT,  -- ms
    execution_time FLOAT,  -- ms

    -- Status
    success BOOLEAN,
    error_code VARCHAR(50),
    error_message TEXT,

    CONSTRAINT valid_timestamp CHECK (timestamp >= '2025-01-01')
);

CREATE INDEX idx_picks_timestamp ON picks(timestamp DESC);
CREATE INDEX idx_picks_robot ON picks(robot_id);
CREATE INDEX idx_picks_success ON picks(success);
CREATE INDEX idx_picks_object_class ON picks(object_class);

-- CALIBRATIONS
CREATE TABLE calibrations (
    calibration_id SERIAL PRIMARY KEY,
    robot_id VARCHAR(50) REFERENCES robots(robot_id),
    timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
    calibration_type VARCHAR(50) NOT NULL,  -- 'hand_eye', 'tcp', 'workspace'
    parameters JSONB NOT NULL,  -- Calibration-specific parameters
    reprojection_error FLOAT,  -- mm (for hand-eye)
    notes TEXT
);
```

```sql
-- SYSTEM_HEALTH
CREATE TABLE system_health (
    health_id SERIAL PRIMARY KEY,
    robot_id VARCHAR(50) REFERENCES robots(robot_id),
    timestamp TIMESTAMP NOT NULL DEFAULT NOW(),

    -- Compute metrics
    cpu_percent FLOAT,
    memory_percent FLOAT,
    disk_percent FLOAT,
    gpu_percent FLOAT,
    gpu_memory_mb FLOAT,

    -- Temperature
    temperature_celsius FLOAT,

    -- Network
    network_latency_ms FLOAT,
    packet_loss_percent FLOAT
);

CREATE INDEX idx_health_timestamp ON system_health(timestamp DESC);
CREATE INDEX idx_health_robot ON system_health(robot_id);

-- ANOMALIES (ML-detected anomalies)
CREATE TABLE anomalies (
    anomaly_id SERIAL PRIMARY KEY,
    robot_id VARCHAR(50) REFERENCES robots(robot_id),
    timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
    anomaly_type VARCHAR(50),  -- 'grasp_failure', 'planning_timeout', 'sensor_drift'
    severity VARCHAR(20),  -- 'low', 'medium', 'high', 'critical'
    description TEXT,
    features JSONB,  -- ML features that triggered detection
    acknowledged BOOLEAN DEFAULT FALSE,
    acknowledged_by VARCHAR(100),
    acknowledged_at TIMESTAMP
);

-- USERS (for authentication)
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(100) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL,  -- 'admin', 'engineer', 'operator', 'viewer'
    created_at TIMESTAMP DEFAULT NOW(),
    last_login TIMESTAMP
```

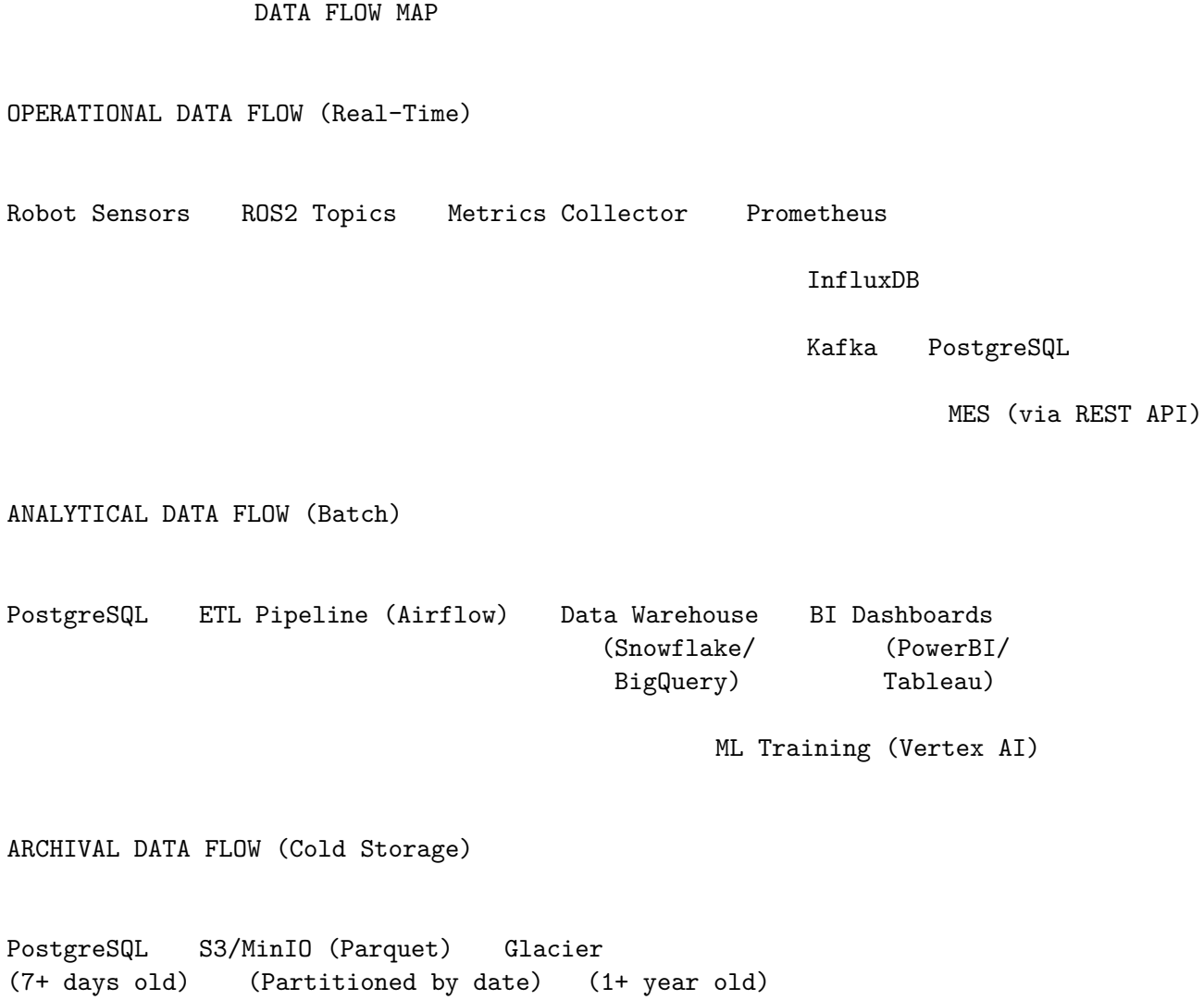
```
);

-- AUDIT_LOGS (immutable, for compliance)
CREATE TABLE audit_logs (
    log_id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
    user_id INT REFERENCES users(user_id),
    action VARCHAR(100) NOT NULL,  -- 'START_SYSTEM', 'STOP_SYSTEM', 'E_STOP', 'CONFIG_CHANGE'
    entity_type VARCHAR(50),  -- 'robot', 'config', 'user'
    entity_id VARCHAR(100),
    details JSONB,
    ip_address INET,
    user_agent TEXT
);

CREATE INDEX idx_audit_timestamp ON audit_logs(timestamp DESC);
CREATE INDEX idx_audit_user ON audit_logs(user_id);
CREATE INDEX idx_audit_action ON audit_logs(action);
```

1.5.3.1 3.3.1 Operational Database (PostgreSQL)

1.5.3.2 3.3.2 Time-Series Database (InfluxDB)

```
// Measurement: cycle_time
// Tags: robot_id, object_class
// Fields: duration (float, seconds)
// Retention: 30 days, downsampled to hourly after 7 days

// Measurement: joint_states
// Tags: robot_id, joint_name (shoulder_pan, shoulder_lift, ...)
// Fields: position (float, rad), velocity (float, rad/s), effort (float, Nm)
// Retention: 7 days (raw), 30 days (1min aggregates)

// Measurement: vision_latency
// Tags: robot_id, stage (detect, pose_estimate)
// Fields: latency_ms (float)
// Retention: 30 days

// Measurement: force_torque
// Tags: robot_id
// Fields: fx, fy, fz (float, N), tx, ty, tz (float, Nm)
// Retention: 7 days (raw), 30 days (10s aggregates)

// Measurement: system_cpu
// Tags: robot_id, host (nuc, jetson)
// Fields: cpu_percent (float)
// Retention: 30 days (1min aggregates)
```

1.5.4 3.4 Data Flow Architecture



1.5.5 3.5 Data Quality Framework

Dimension	Definition	Measurement	Target	Actions
Accuracy	Correctness of data values	% of picks with pose error <1mm	>99%	Calibration checks, outlier detection
Completeness	All required fields populated	% of picks with non-null critical fields	100%	Validation rules, default values
Consistency	Data agrees across systems	% match between robot logs and MES	>99.5%	Reconciliation jobs, checksums

Dimension	Definition	Measurement	Target	Actions
Timeliness	Data available when needed	Latency: sensor → dashboard	<5 sec	Stream processing, caching
Uniqueness	No duplicate records	Duplicate pick_ids	0	Primary key constraints, deduplication
Validity	Data conforms to rules	% of values within valid ranges	100%	Check constraints, input validation

1.5.6 3.6 Data Governance

1.5.6.1 3.6.1 Data Ownership

Data Domain	Data Owner	Data Steward	Responsibilities
Robot Operational Data	Engineering Manager	Senior Robot Engineer	Schema design, quality monitoring, access control
Production Metrics	Operations Manager	Production Supervisor	KPI definitions, reporting requirements
Quality Data	QA Manager	QA Lead	Defect tracking, root cause analysis
Financial Data	CFO	Finance Analyst	Cost tracking, ROI calculations

1.5.6.2 3.6.2 Data Lineage Example

```
DATA LINEAGE: picks.cycle_time


Source:
  Robot Controller (ros2_control) @ 1000 Hz


  ROS2 Topic: /task/status (10 Hz)
     Published by: task_orchestrator_node
     Message type: robot_msgs/TaskStatus
     Field: cycle_time (float32)


  Metrics Collector (Python) @ 10 Hz
     Subscribes to: /task/status
     Aggregates: mean, p95, p99 per minute

      InfluxDB: cycle_time measurement
```

```
        Retention: 30 days
        Downsampling: 1min → 1hour after 7 days

       PostgreSQL: picks.cycle_time column
        Persisted per pick
        Indexed for queries
        Partition by month
```

Consumers:

- Grafana Dashboard (real-time chart)
- PowerBI Report (daily aggregates)
- ML Model (anomaly detection)
- MES API (production tracking)

1.5.7 3.7 Master Data Management (MDM)

1.5.7.1 3.7.1 Master Data Entities

Entity	Source of Truth	Sync Frequency	Attributes
Robot	Robot Registry DB	Real-time	robot_id, model, serial, location, IP
Object Classes	Vision ML Registry	On model update	class_name, dimensions, weight, image
Workspace Zones	Configuration DB	On change	zone_id, type (pick/place), bounds
Users	Identity Provider (Keycloak)	Real-time (SSO)	user_id, email, role, permissions

---

1.6 4. Integration Architecture

1.6.1 4.1 Integration Landscape

```
                INTEGRATION ARCHITECTURE


EXTERNAL SYSTEMS                  ROBOT SYSTEM



   MES        REST API       Web Backend
                (JSON/HTTPS)       (FastAPI)


                                        ROS2 Bridge
```

```
   ERP        REST API
               (JSON/HTTPS)

                                    ROS2 Middleware
                                (DDS)
  SCADA       OPC-UA
               (real-time)



  BI/          JDBC          PostgreSQL
Analytics       (SQL queries)
```

1.6.2 4.2 Integration Patterns

1.6.2.1 4.2.1 Request-Reply (Synchronous)

Use Case: Configuration changes, manual commands

```
# Example: Start system via REST API

# Client (MES)
import requests

response = requests.post(
    'https://robot.factory.com/api/system/start',
    headers={'Authorization': 'Bearer <token>'},
    json={'robot_id': 'robot_01'}
)

if response.status_code == 200:
    print(f"System started: {response.json()}")
else:
    print(f"Error: {response.json()['error']}")

# Server (FastAPI)
from fastapi import FastAPI, HTTPException

@app.post("/api/system/start")
async def start_system(request: StartRequest):
    # Call ROS2 service
    ros_client = node.create_client(StartSystem, '/system/start')
    ros_response = await ros_client.call_async(StartSystem.Request())

    if ros_response.success:
        return {"success": True, "message": "System started"}
    else:
        raise HTTPException(status_code=500, detail=ros_response.message)
```

1.6.2.2 4.2.2 Publish-Subscribe (Asynchronous) **Use Case:** Real-time telemetry, events

```
# Example: Publish pick events to Kafka

# Producer (Metrics Collector)
from kafka import KafkaProducer
import json

producer = KafkaProducer(
    bootstrap_servers=['kafka:9092'],
    value_serializer=lambda v: json.dumps(v).encode('utf-8')
)

def on_pick_complete(pick_data):
    producer.send('picks', value=pick_data)
    producer.flush()

# Consumer (MES Integration Service)
from kafka import KafkaConsumer

consumer = KafkaConsumer(
    'picks',
    bootstrap_servers=['kafka:9092'],
    value_deserializer=lambda m: json.loads(m.decode('utf-8'))
)

for message in consumer:
    pick = message.value
    update_mes_production_count(pick['robot_id'], pick['success'])
```

1.6.2.3 4.2.3 Event-Driven (Reactive) **Use Case:** Error handling, alerts

```
# CloudEvents specification for robot error

{
  "specversion": "1.0",
  "type": "com.factory.robot.error",
  "source": "/robots/robot_01",
  "id": "e12345-abcdef",
  "time": "2025-10-18T10:23:45.123Z",
  "datacontenttype": "application/json",
  "data": {
    "error_code": "GRASP_FAILURE",
    "severity": "medium",
    "message": "Object slipped during grasp",
    "retry_count": 2
  }
}
```

1.6.3 4.3 API Specifications

1.6.3.1 4.3.1 REST API (External Integration)

Base URL: `https://robot.factory.com/api/v1`
Authentication: OAuth2 Bearer Token (JWT) **Rate Limit:** 1000 requests/hour per API key

Endpoint	Method	Description	Auth	Rate Limit
`/system/status`	GET	Get system status	Required	60/min
`/system/start`	POST	Start robot operation	Admin/Engineer	10/min
`/system/stop`	POST	Stop robot operation	Admin/Engineer	10/min
`/picks`	GET	Query pick history	Required	60/min
`/picks/{id}`	GET	Get pick details	Required	100/min
`/metrics`	GET	Get performance metrics	Required	60/min
`/config`	GET	Get configuration	Engineer	30/min
`/config`	PUT	Update configuration	Admin	10/min
`/calibration`	POST	Start calibration	Engineer	5/min

Example Response:

```
GET /api/v1/system/status

{
  "robot_id": "robot_01",
  "status": "running",
  "uptime_seconds": 3456789,
  "picks_today": 28340,
  "success_rate": 0.984,
  "current_cycle_time": 1.82,
  "errors_today": 45,
  "last_error": {
    "timestamp": "2025-10-18T10:23:45Z",
    "code": "GRASP_RETRY",
    "resolved": true
  }
}
```

1.6.3.2 4.3.2 gRPC API (Internal High-Performance)

Service Definition (Protocol Buffers):

```
syntax = "proto3";

package robot.control;

service RobotController {
  rpc GetJointStates(Empty) returns (JointStates);
  rpc MoveToJointPositions(JointPositions) returns (MoveResponse);
  rpc ExecutePickPlace(PickPlaceRequest) returns (PickPlaceResponse);
  rpc StreamMetrics(Empty) returns (stream Metrics);
```

```
}

message JointStates {
  repeated double positions = 1;  // radians
  repeated double velocities = 2;  // rad/s
  repeated double efforts = 3;  // Nm
  int64 timestamp = 4;  // Unix timestamp (ns)
}

message PickPlaceRequest {
  Pose pick_pose = 1;
  Pose place_pose = 2;
  float approach_distance = 3;  // meters
  float gripper_force = 4;  // Newtons
}

message PickPlaceResponse {
  bool success = 1;
  string message = 2;
  float execution_time = 3;  // seconds
}

message Pose {
  float x = 1;
  float y = 2;
  float z = 3;
  float qx = 4;  // quaternion
  float qy = 5;
  float qz = 6;
  float qw = 7;
}
```

1.6.4 4.4 Message Formats

```
{
  "event_type": "pick_completed",
  "timestamp": "2025-10-18T10:23:45.123Z",
  "robot_id": "robot_01",
  "pick_id": 127,
  "object": {
    "class": "red_cube",
    "confidence": 0.982,
    "pose": {
      "position": {"x": 0.452, "y": 0.123, "z": 0.234},
      "orientation": {"qx": 0.01, "qy": -0.02, "qz": 0.03, "qw": 0.999}
    }
```

```
  },
  "grasp": {
    "quality": 0.87,
    "force": 25.3
  },
  "performance": {
    "cycle_time": 1.82,
    "vision_latency": 42,
    "planning_time": 187,
    "execution_time": 762
  },
  "success": true
}
```

1.6.4.1 4.4.1 Pick Event (Kafka/MQTT)

```
{
  "alert_type": "system_error",
  "severity": "high",
  "timestamp": "2025-10-18T10:23:45Z",
  "robot_id": "robot_01",
  "title": "Planning Timeout Threshold Exceeded",
  "description": "Motion planning took 512ms, exceeding 500ms threshold",
  "impact": "Potential cycle time increase",
  "recommended_action": "Check planner configuration, consider switching to RRTConnect",
  "affected_picks": [125, 126, 127],
  "dashboard_link": "https://robot.factory.com/dashboard?robot=robot_01&time=2025-10-18T10:23:4
}
```

1.6.4.2 4.4.2 System Alert (Email/Slack)

1.6.5 4.5 Integration Governance

Aspect	Policy	Enforcement
Versioning	Semantic versioning (v1.2.3) for all APIs	API version in URL path
Deprecation	6-month notice, maintain 2 versions concurrently	Deprecation warnings in responses
Documentation	OpenAPI 3.0 spec for all REST APIs	Auto-generated Swagger UI
Testing	Contract testing for all integrations	Pact.io, automated in CI/CD
Monitoring	Track latency, error rate, throughput	Prometheus + Grafana
Security	TLS 1.3, OAuth2, API keys	Kong API Gateway

1.7 5. Business Architecture

1.7.1 5.1 Value Stream Map

VALUE STREAM: Pick-and-Place Manufacturing Operation

Receive Objects	Identify Objects	Pick & Grasp	Place & Release
Manual (Warehouse)	AI Vision (30 FPS, <50ms)	Robotic (99%+ success) (0.8s execution)	Robotic (±0.1mm accuracy) (0.3s execution)
Lead Time Variable	Cycle Time 0.05s	Cycle Time 0.8s	Cycle Time 0.3s

Quality Inspection	Verify Placement	Log Data (Automated)
(Optional) Manual/Auto	Camera (real-time)	Real-time to PostgreSQL/MES

METRICS:

- Total Lead Time: 1.15s (vs 4.2s manual)
- Process Cycle Efficiency: 95% (value-add time / total time)
- First-Pass Yield: 98.4% (vs 95% manual)
- Cost per Pick: $0.003 (vs $0.012 manual)

1.7.2 5.2 Business Capability Model

BUSINESS CAPABILITY MODEL

LEVEL 1: Strategic Capabilities

- Production Planning & Scheduling
- Quality Management
- Asset Performance Management
- Workforce Management

LEVEL 2: Core Capabilities (Robot-Enabled)

Object Detection	Grasp Planning	Motion Execution
• Vision AI	• Force closure	• Path planning
• Pose estimation	• Collision check	• Trajectory ctrl
• Classification	• Quality scoring	• Safety monitor

LEVEL 3: Supporting Capabilities

- System Monitoring & Alerting
- Data Collection & Analytics
- Calibration & Maintenance
- Security & Access Control

CAPABILITY MATURITY:

Level 5: Optimizing	• Object Detection (AI-driven)
Level 4: Managed	• Grasp Planning, Motion Execution
Level 3: Defined	• Monitoring, Data Analytics
Level 2: Repeatable	• Calibration
Level 1: Initial	• Predictive Maintenance (roadmap)

1.7.3 5.3 Operating Model

1.7.3.1 5.3.1 Organization Structure

Plant Manager

Production Manager | Engineering Manager | Quality Manager

Operators (3 shifts) | Robot Engineers (Software/Mech) | QA Inspectors

```
Data Scientist   Maintenance
(AI/ML)          Technician
```

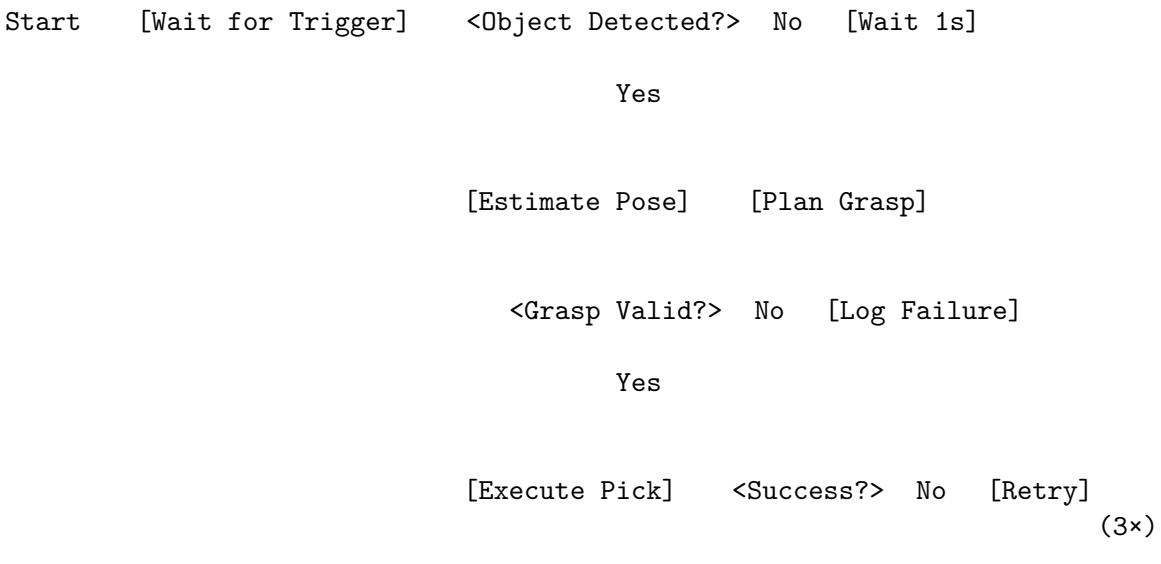
1.7.3.2 5.3.2 RACI Matrix

Activity	Operator	Engineer	Manager	QA	Maintenance
Daily Operations	R	C	I	I	I
System Configuration	I	R, A	C	I	I
Troubleshooting	I	R, A	C	I	C
Calibration	I	R	A	C	C
Performance Analysis	I	C	R, A	I	I
AI Model Updates	I	R	A	I	I
Preventive Maintenance	I	C	I	I	R, A
Quality Audits	I	I	C	R, A	I

R=Responsible, A=Accountable, C=Consulted, I=Informed

1.7.4 5.4 Business Process Models (BPMN)

1.7.4.1 5.4.1 Pick-and-Place Process



```
                    Yes

          [Execute Place]    <Success?>  No

                    Yes

          [Log Success]    End
```

1.7.5 5.5 KPI Dashboard (Business Metrics)

KPI Category	Metric	Target	Current	Trend
Productivity	Picks per Day	28,000	28,340	+1.2%
Quality	Success Rate	>98%	98.4%	+0.6%
Efficiency	Cycle Time	<2.0s	1.82s	+9%
Reliability	Uptime	>99%	99.7%	+0.2%
Financial	Cost per Pick	<$0.005	$0.003	+40%
Safety	Incidents	0	0	Stable

1.8 6. Cross-Architecture Alignment

1.8.1 6.1 Architecture Alignment Matrix

Business Capability	Application	Data	Integration	Technology
Object Detection	Vision Pipeline (YOLOv8)	object_classes (MDM), detections (event)	ROS2 topic, Kafka event	Python, PyTorch, TensorRT, Jetson Xavier
Grasp Planning	Grasp Synthesizer	grasp candidates (transient)	ROS2 service	Python, NumPy, Open3D
Motion Execution	MoveIt2, ros2_control	joint trajectories (transient)	ROS2 action	C++, OMPL, RT-Linux
Data Analytics	Grafana, PowerBI	picks (PostgreSQL), metrics (InfluxDB)	REST API, JDBC	PostgreSQL, InfluxDB, Grafana

1.8.2 6.2 Architecture Decision Alignment

Every architecture decision is validated against all four perspectives:

Example: Choice of ROS2 Humble

Perspective	Validation	Status
Enterprise	Aligns with open-standards principle, large ecosystem	Approved
Data	Supports efficient message serialization (CDR), DDS discovery	Approved
Integration	Native pub/sub, service, action patterns for interoperability	Approved
Business	Enables rapid development (shorter time-to-market), cost-effective	Approved

---

1.9 7. Governance & Standards

1.9.1 7.1 Architecture Review Board (ARB)

Purpose: Ensure architectural consistency, quality, and alignment with strategy.

Members: - Enterprise Architect (Chair) - Data Architect - Integration Architect - Business Architect - CTO (Observer)

Responsibilities: - Review architecture decision records (ADRs) - Approve deviations from standards - Maintain architecture roadmap - Conduct quarterly architecture health checks

1.9.2 7.2 Architecture Health Metrics

Metric	Definition	Target	Current
Standard Compliance	% of systems following EA standards	>95%	98%
Technical Debt	Effort to remediate non-standard implementations	<5% of total dev effort	3.2%
Integration Complexity	# of point-to-point integrations / total integrations	<20%	12%
Data Quality Score	Composite score (accuracy, completeness, timeliness)	>95%	97.3%
API SLA Adherence	% of API calls meeting latency SLA (<100ms)	>99%	99.6%

1.10 Summary

1.10.1 Multi-Architecture Coverage

Enterprise Architecture: Strategic alignment, standards, roadmap (3-year) **Data Architecture:** Logical/physical models, governance, quality, lineage **Integration Architecture:** Patterns, APIs (REST/gRPC), message formats **Business Architecture:** Value streams, capabilities, operating model, KPIs

1.10.2 Key Deliverables

Architecture	Artifacts	Governance
Enterprise	EA principles, technology standards, application portfolio, roadmap	ARB review, quarterly health checks
Data	Conceptual/logical/physical models, data lineage, quality rules, MDM	Data governance council, monthly audits
Integration	API specs (OpenAPI), message schemas (Protobuf/JSON), integration patterns	API review process, contract testing
Business	Value stream maps, capability model, BPMN processes, KPI dashboard	Business review, monthly KPI tracking

---

Document Status: v1.0 Complete **Related Documents:** Technical Stack (05), HLD (08), LLD (14) **Framework:** TOGAF 9.2, ArchiMate 3.1
