### Updated Version of Orgo v2 Blueprint Section 6 - Workflow Integration

#### Section 6: Workflow Integration

This section outlines the design, implementation, and management of workflows in Orgo. Updates include a shift to modularized and metadata-driven workflow logic, ensuring scalability, efficiency, and adaptability to diverse organizational needs.

### 6.1 Purpose of Workflow Integration

Objective:

* Define and implement structured workflows for automated task management, traceability, and compliance.
* Transition from domain-specific workflows to a dynamic, metadata-driven system.

Outcome:

* Generalized workflows adaptable across domains based on task attributes.
* Streamlined task creation, routing, and escalation processes.

### 6.2 Workflow Design Principles

1. Metadata-Driven Logic:

* Workflows rely on task attributes, such as type, metadata, and priority, for routing and execution.
* Example: A maintenance task with metadata.subtype=plumbing triggers plumbing-specific logic within a generalized task handler.

2. Modularity:

* Workflows are designed using YAML or JSON templates for easy customization and reusability.

3. Role-Based Routing:

* Tasks are dynamically routed to individuals or teams based on metadata and predefined rules, ensuring continuity during personnel changes.

4. Automated Attachments:

* Dynamic Attachments: Generated based on keywords or metadata (e.g., attaching a safety protocol for "leak").
* Static Attachments: Standard documents included for specific task types.

5. Feedback Loop Integration:

* Responses from assigned personnel update task statuses or trigger additional actions, such as escalation.

6. Scalability:

* Workflow components handle high volumes of tasks efficiently, leveraging centralized task management and database-driven logic.

### 6.3 Core Workflow Components

1. Trigger:

* Workflows are initiated by an event, such as an incoming email, a manual entry, or a system alert.

2. Parsing:

* Metadata (e.g., keywords, sender information) is extracted and analyzed to classify the task.

3. Routing:

* The rule engine dynamically applies conditions to determine task assignment based on attributes like type and metadata.

4. Action:

* Tasks are executed with predefined templates, notifications, or document attachments.

5. Escalation:

* Unresolved tasks escalate to higher authorities based on predefined timeframes or conditions.

6. Resolution:

* Upon completion, tasks are logged, and a summary is sent to relevant stakeholders.

### 6.4 Workflow Examples

1. Maintenance Request Workflow:

* Trigger: An email reports a maintenance issue.
* Parsing: Keywords such as "leak" and "urgent" are identified.
* Routing: The task is routed to the facilities team based on type=maintenance and metadata.subtype=plumbing.
* Attachments: Location map and resolution protocol are included.
* Escalation: If unresolved within 2 hours, escalate to the supervisor.
* Resolution: Task status is updated, and a summary is logged.

2. HR Issue Reporting Workflow:

* Trigger: An employee submits a grievance to HR.
* Anonymization: Identifying metadata is stripped to protect privacy.
* Routing: The task is routed to HR and flagged for immediate action.
* Attachments: Relevant policies and reporting templates are included.
* Escalation: If not addressed within 48 hours, escalate to the HR manager.

3. IT Support Workflow:

* Trigger: A system outage report is received.
* Parsing: Keywords like "outage" and "system failure" are extracted.
* Routing: The task is assigned to the IT support team.
* Attachments: Logs of affected systems are included for troubleshooting.
* Resolution: IT team resolves the issue or escalates to infrastructure specialists.

### 6.5 Escalation Rule Standardization

Conditions:

* Escalation is triggered when tasks remain unresolved beyond defined timeframes or based on priority levels.

Actions:

* Notify higher-level personnel or escalate the task to supervisors, managers, or leadership.

Levels:

* Primary: Notify the task owner or team lead.
* Secondary: Notify the department manager.
* Final: Escalate to senior management.

### 6.6 Workflow Testing and Validation

1. Unit Testing:

* Validate individual workflow components, such as parsing and routing logic.

2. End-to-End Testing:

* Simulate complete workflows to ensure accuracy in task routing, attachment handling, and escalation.

3. Stress Testing:

* Test workflow performance under high task volumes to confirm scalability and reliability.

4. Validation Reports:

* Document results of testing to ensure workflows meet organizational requirements.

### 6.7 Deliverables

1. YAML and JSON Templates:
   * Predefined rules for common workflows, such as maintenance, HR, and IT tasks.
2. Logs and Audit Trails:
   * Examples of routing decisions, escalations, and task resolutions.
3. Testing Framework:
   * Tools and processes for validating workflows before deployment.

### Summary

This section redefines Orgo’s workflow integration by transitioning from domain-specific logic to a centralized, metadata-driven approach. The dynamic workflows outlined here ensure efficiency, scalability, and adaptability, allowing organizations to handle diverse task types seamlessly. By leveraging modular templates, role-based routing, and robust escalation mechanisms, Orgo provides a framework for streamlined and reliable task management.