Supplementary Materials

This supplement provides the complete extraction schema (Listing 1) referenced in the main paper.

1 Pydantic Extraction Schema (Listing 1)

The schema is implemented as a hierarchy of Pydantic v2 data models. The root model `WorkOrderECMRecord` comprises three optional list fields corresponding to the three schema layers: `entities` (`List[ECMEntity]`), `change_events` (`List[ChangeEvent]`), and `traceability_links` (`List[TraceabilityLink]`). Each leaf model field carries a `description` argument; these descriptions are included in the JSON Schema serialization of the model and are thereby visible to the model during inference, providing per-slot semantic grounding.

Listing 1: Complete Pydantic v2 schema defining `WorkOrderECMRecord` and its component models.

```
"""
Listing 1  Complete Pydantic v2 extraction schema for ECM traceability.

This is the WorkOrderECMRecord schema referenced in the paper. Every field carries
a `description`, which is included in the JSON Schema serialization
(`WorkOrderECMRecord.model_json_schema()`) and thereby made visible to the LLM
during inference. The same schema is used by the extraction pipeline, the Ollama
Modelfile system prompt, and the evaluation harness (evaluate_ecm.py).

Requires: pydantic >= 2
"""
from __future__ import annotations
from enum import Enum
from typing import List, Optional
from pydantic import BaseModel, Field


# ---------------------------------------------------------------- enumerations
class EntityType(str, Enum):
    """Typed information objects that change events operate on or refer to."""
    component = "Component"
    specification = "Specification"
    regulation = "Regulation"
    exception_ticket = "ExceptionTicket"
    responsible_party = "ResponsibleParty"


class EventType(str, Enum):
    """Bounded occurrences in which the state of one or more entities is altered."""
    substitution = "Substitution"
    specification_revision = "SpecificationRevision"
    quantity_adjustment = "QuantityAdjustment"
    status_transition = "StatusTransition"
```

```python


class LinkType(str, Enum):
    """Explicit cross-record relationships needed for audit and impact analysis."""
    cross_order_reference = "CrossOrderReference"
    exception_ticket_link = "ExceptionTicketLink"


# ------------------------------------------------------------ leaf models
class ECMEntity(BaseModel):
    """A single typed ECM entity extracted from the annotation."""
    entity_type: EntityType = Field(
        ..., description="The ECM entity category, drawn from the EntityType enum.")
    value: str = Field(
        ..., description="The entity string exactly as it appears in the source "
                         "annotation (part number, specification, standard identifier, "
                         "ticket id, or responsible party). Do not paraphrase or expand.")
    is_truncated: bool = Field(
        default=False,
        description="True if the entity string appears cut off by the 255-character "
                    "Access field limit (i.e., the visible text ends mid-token).")


class ChangeEvent(BaseModel):
    """A single engineering-change event extracted from the annotation."""
    event_type: EventType = Field(
        ..., description="The change-event category, drawn from the EventType enum.")
    entities: List[str] = Field(
        default_factory=list,
        description="The entity values involved in this event (e.g., the components or "
                    "specifications being changed), each copied verbatim from the source.")
    from_value: Optional[str] = Field(
        default=None,
        description="The prior value before the change (e.g., the original quantity, "
                    "specification, or component), if stated in the text; otherwise null.")
    to_value: Optional[str] = Field(
        default=None,
        description="The new value after the change, if stated in the text; otherwise null.")
    reason: Optional[str] = Field(
        default=None,
        description="A brief reason or justification for the change, if stated; otherwise "
                    "null. This field may paraphrase the source.")
    is_truncated: bool = Field(
        default=False,
        description="True if the event description appears cut off by the 255-character "
                    "limit (e.g., a substitution begun but not completed in visible text).")


class TraceabilityLink(BaseModel):
    """A single cross-record traceability link extracted from the annotation."""
    link_type: LinkType = Field(
        ..., description="The link category, drawn from the LinkType enum.")
    source_id: Optional[str] = Field(
        default=None,
        description="Identifier of the current work order, if explicitly stated; "
                    "otherwise null.")
    target_id: str = Field(
        ..., description="Identifier of the referenced work order or exception record, "
                         "copied verbatim from the source annotation.")
```

```python


# ------------------------------------------------------------------ root model
class WorkOrderECMRecord(BaseModel):
    """Root extraction model. The structured target produced for each work-order
    annotation, comprising the three schema layers."""
    entities: List[ECMEntity] = Field(
        default_factory=list,
        description="All ECM entities mentioned in the annotation.")
    change_events: List[ChangeEvent] = Field(
        default_factory=list,
        description="All change events described in the annotation.")
    traceability_links: List[TraceabilityLink] = Field(
        default_factory=list,
        description="All cross-order references and exception-ticket links in the "
                    "annotation.")


if __name__ == "__main__":
    # Emit the JSON Schema serialization that is supplied to the LLM at inference time.
    import json
    print(json.dumps(WorkOrderECMRecord.model_json_schema(), indent=2, ensure_ascii=False))
```