

```
{
  "orchestrator": {
    "name": "metaphor→meta-metaphor.orchestrator",
    "describe_how_nodes_interact": [
      "1) Perception→Anchoring: captar señales del objeto (texto/código/escena) y convertirlas en anchors con evidencia y peso.",

      "2) Esqueleto (schema): a partir de anchors, construir la estructura interna (roles, composición, vecindad, orden).",

      "3) Vecindad compartida: verificar vocabulario mínimo común entre dominio_origen y dominio_destino.",

      "4) Alineación estructural: emparejar subestructuras homólogas (structure-mapping) preservando composición y vecindad.",

      "5) Reglas de mapeo: sintetizar metaphor_map (adapter) desde origen→destino con constraints de preservación.",

      "6) Evaluación: medir poder predictivo del mapping sobre datos reales (¿predice imports, rutas, dependencias?).",

      "7) Emisión ejecutable: producir artefactos (anchors.jsonl, edges.jsonl, metaphor_map.json) y el integration_profile.",

      "8) Bucle de mejora: si baja el poder predictivo o falta vecindad, ajustar vocabulario/anchors/constraints y reintentar."
    ],
    "policy": {
      "evidence_priority": ["perception>textual", "estructura>palabras", "hechos>asociaciones"],
      "constraints_preserved": ["composition", "adjacency", "ordering", "type-role"],
      "provenance": true
    },
    "nodes": [
      { "id": "meta_language_parser",
        "name": "metaphor→meta-metaphor.orchestrator"
      }
    ]
  }
}
```

```
"what": "Detecta marcadores meta ([]), MAYÚSCULAS, etiquetas) para crear anclas iniciales.",  
    "inputs": ["texto crudo"],  
    "outputs": ["anchors:meta"]  
,  
    { "id": "perception_capture",  
        "what": "Modelo general de captura multimodal (en código: texto, patrones sintácticos, nombres de carpeta).",  
        "inputs": ["corpus(JSONL) o texto"],  
        "outputs": ["anchors:perception"]  

```

```
"what": "Alinea subestructuras homólogas (structure-mapping).",
"inputs": ["schema_origen", "schema_destino", "alias_map"],
"outputs": ["candidate_mappings"]
},
{ "id": "mapping_constraints",
"what": "Define qué propiedades deben preservarse (composición, vecindad, orden).",
"inputs": ["candidate_mappings"],
"outputs": ["validated_mappings"]
},
{ "id": "adapter_synthetizer",
"what": "Genera metaphor_map (reglas ejecutables from→to con preservación).",
"inputs": ["validated_mappings"],
"outputs": ["metaphor_map.json"]
},
{ "id": "predictive_evaluator",
"what": "Mide poder predictivo del mapeo sobre casos reales.",
"inputs": ["metaphor_map.json", "schema_destino", "corpus_destino"],
"outputs": ["fitness_score", "diagnostics"]
},
{ "id": "provenance_tracker",
"what": "Adjunta trazabilidad a anchors/edges/mappings.",
"inputs": ["anchors", "edges", "metaphor_map.json"],
"outputs": ["provenance_annotations"]
},
{ "id": "fractal_composer",
"what": "Compone nodos en jerarquías (nodo↔subnodos) y permite reutilización."
},
```

```
"inputs": ["schema", "edges"],  
"outputs": ["graph:composed"]  
},  
{ "id": "narrative_situator",  
  "what": "Crea ejemplos situados (pizza/restaurante) para transferir estructura entre  
dominios.",  
  "inputs": ["graph:composed"],  
  "outputs": ["cases:didácticos"]  
},  
{ "id": "exporter_SOT",  
  "what": "Single Source of Truth: extrae path+content_raw a JSONL (cuando el dominio es  
código).",  
  "inputs": ["directorio_proyecto"],  
  "outputs": ["corpus.jsonl"]  
},  
{ "id": "edge_builder",  
  "what": "Genera edges desde el corpus (imports→composition, rutas→adjacency).",  
  "inputs": ["corpus.jsonl", "anchors"],  
  "outputs": ["edges"]  
},  
{ "id": "alias_resolver",  
  "what": "Concilia nombres distintos para el mismo rol (ingrediente↔dependencia).",  
  "inputs": ["lexicon_origen", "lexicon_destino"],  
  "outputs": ["alias_map"]  
},  
{ "id": "integration_profile_emitter",  
  "what": "Formaliza Always-Integrable: provides/requires/constraints/types + adapters."}
```

```
"inputs": ["metaphor_map.json", "schema", "edges", "provenance"],  
"outputs": ["integration_profile"]  
},  
{ "id": "metaphor_definer",  
"what": "Formaliza 'metáfora' como transferencia de estructura entre dominios vecinos.",  
"inputs": ["schema_origen", "schema_destino"],  
"outputs": ["metaphor_spec"]  
},  
{ "id": "meta_metaphor_formalizer",  
"what": "Eleva la metáfora a especificación ejecutable (metaphor_map como adapter).",  
"inputs": ["metaphor_spec", "validated_mappings"],  
"outputs": ["adapter_spec (meta-metáfora ejecutable)"]  
}  
]
```