시나리오 합성 과정

0) 준비물(입력 아티팩트 4종)

(A) QueryGoal.json — *교수님 형식 그대로, 시간은 매크로*

```
{
  "QueryGoal": {
    "goalId": "goal-job-eta-0001",
    "goalType": "predict_job_completion_time",
    "parameters": [
      { "key": "jobId",   "value": "JOB-7f2e3a8b-1d" },
      { "key": "dueDate", "value": "@현재시간" }
    ],
    "outputSpec": [
      { "name": "completion_time", "datatype": "datetime" },
      { "name": "tardiness_s",     "datatype": "number" },
      { "name": "sla_met",         "datatype": "boolean" }
    ],
    "termination": [
      { "key": "condition", "value": "on_job_completed" },
      { "key": "timeout",   "value": "PT4H" }
    ],
    "selectedModelRef": "aas://ModelCatalog/JobETAModel@1.4.2",
    "selectedModel": {
      "modelId": "JobETAModel",
      "MetaData": "JobEtaMetaData.json",
      "outputs": ["completion_time","tardiness_s","sla_met"],
      "preconditions": { "units.time": "s", "runtime.freshness": "PT30S" },
      "container": { "image": "registry/factory/job-eta:v1.4.2", "digest": "sha2
56:..." },
      "catalogVersion": "1.4.2",
      "frozenAt": "@현재시간"
    },
```

```
    "selectionProvenance": {
      "ruleName": "SWRL:Goal2JobETA",
      "ruleVersion": "v1.0",
      "engine": "SWRL",
      "evidence": { "matched": ["goalType==predict_job_completion_time","p
urpose==DeliveryPrediction"] },
      "inputsHash": "sha256:...",
      "timestamp": "2025-09-12T00:00:00Z",
      "notes": ""
    }
  }
}
```

(B) JobEtaMetaData.json — *모델 메타데이터(필수 입력의 진실원천)*

```
{
  "modelId": "JobETAModel",
  "requiredInputs": ["JobRoute","MachineState","Calendar","SetupMatri
x","WIP","Backlog"],
  "preconditions": { "units.time": "s", "runtime.freshness": "PT30S" },
  "outputs": ["completion_time","tardiness_s","sla_met"]
}
```

(C) bindings.job-eta.yaml — *바인딩/글롭/결합 규칙(결정적 의미론 고정)*

```
schema: 1
sources:
  JobRoute:
    uri: "aas://FactoryTwin/JobRoute/{jobId}"
  MachineState:
    uri: "aas://FactoryTwin/State/Machine?at={bindAt}"
    defaults: { bindAt: "@현재시간" }
  Calendar:
    uri: "file://config/factory_calendar.yaml"
  SetupMatrix:
    uri: "file://config/setup_matrix.yaml"
```

```
  WIP:
    uri: "file://state/wip/*.json"
    glob:
      sort: ["mtime:desc","name:asc"]  # 결정적 정렬
      window: { count: 5 }            # 최신 5개만
    combine:
      op: "overlay"                   # 충돌은 last-wins
      key: "partId"
  Backlog:
    uri: "file://state/backlog/*.json"
    glob:
      sort: ["mtime:desc","name:asc"]
      window: { since: "2025-09-01T00:00:00+09:00", until: "@현재시간" }
    combine:
      op: "concat"
```

(D) `job-eta.scenario.yaml.j2` — *렌더 템플릿*

```
job:
  id: "{{ params.jobId }}"
  due: "{{ params.dueDate }}"
inputs:
  job_route: "{{ bindings.JobRoute }}"
  machine_state: "{{ bindings.MachineState }}"
  calendar: "{{ bindings.Calendar }}"
  setup_matrix: "{{ bindings.SetupMatrix }}"
  wip_files: {{ bindings.WIP | tojson }}
  backlog_files: {{ bindings.Backlog | tojson }}
runtime:
  freshness_requirement: "{{ meta.preconditions['runtime.freshness'] }}"
```

---

1) 전처리 — 실시간 시간 치환(@현재시간)

실행 시, 생성기는 매크로를 확정합니다(Asia/Seoul 기준).

치환 결과(발췌):

```
- "dueDate": "@현재시간"
+ "dueDate": "2025-09-12T15:13:31+09:00"

- "frozenAt": "@현재시간"
+ "frozenAt": "2025-09-12T15:13:31+09:00"

- defaults: { bindAt: "@현재시간" }
+ defaults: { bindAt: "2025-09-12T15:13:31+09:00" }
```

이후 Reasoner/SW(R)L/SHACL에는 **순수 값**만 전달됩니다.

---

2) 메타 로드 — requiredInputs 확보

생성기는 `selectedModel.MetaData`를 로드하여 **필수 입력**을 얻습니다:

```
["JobRoute","MachineState","Calendar","SetupMatrix","WIP","Backlog"]
```

이 목록이 **검증·조립의 기준**이 됩니다.

---

3) 바인딩 확장 — URI/글롭/윈도우/정렬 적용

- `parameters.jobId = JOB-7f2e3a8b-1d`
- `bindings.JobRoute` → `aas://FactoryTwin/JobRoute/JOB-7f2e3a8b-1d`
- `bindings.MachineState` → `aas://FactoryTwin/State/Machine?at=2025-09-12T15:13:31+09:00`
- `WIP` 글롭 결과(예시 mtime 내림차순):

  ```
  file://state/wip/wip_2025-09-12T15-10.json
  file://state/wip/wip_2025-09-12T15-05.json
  file://state/wip/wip_2025-09-12T15-00.json
  file://state/wip/wip_2025-09-12T14-55.json
  file://state/wip/wip_2025-09-12T14-50.json
  file://state/wip/wip_2025-09-12T14-45.json
  ```

  `window.count=5` → **최신 5개**만 채택(14:45 파일 제외).

- `Backlog` 글롭 결과는 9월 1일~현재까지 범위로 `concat`.

---

4) 결합(Combine) — overlay / concat의 결정적 의미론

- **WIP (overlay, key=partId)**

  서로 다른 파일에 같은 `partId`가 있을 경우 **더 최신 파일의 필드가 덮어씀(last-wins)**.

  ```
  // 예: 두 파일에 공통 partId="P-101"
  // 이전 파일: {"partId":"P-101","qty":5,"prio":"normal"}
  // 최신 파일: {"partId":"P-101","qty":6}
  // overlay 결과: {"partId":"P-101","qty":6,"prio":"normal"}
  ```

- **Backlog (concat)**

  시계열/리스트는 정렬 후 단순 연결. 충돌 정의가 없으므로 **그대로 이어붙임**.

---

5) 렌더 — 최종 scenario.yaml 산출

```
# ./scenarios/goal-job-eta-0001.yaml
job:
  id: "JOB-7f2e3a8b-1d"
  due: "2025-09-12T15:13:31+09:00"
inputs:
  job_route: "aas://FactoryTwin/JobRoute/JOB-7f2e3a8b-1d"
  machine_state: "aas://FactoryTwin/State/Machine?at=2025-09-12T15:13:3
1+09:00"
  calendar: "file://config/factory_calendar.yaml"
  setup_matrix: "file://config/setup_matrix.yaml"
  wip_files:
    - "file://state/wip/wip_2025-09-12T15-10.json"
    - "file://state/wip/wip_2025-09-12T15-05.json"
    - "file://state/wip/wip_2025-09-12T15-00.json"
    - "file://state/wip/wip_2025-09-12T14-55.json"
    - "file://state/wip/wip_2025-09-12T14-50.json"
  backlog_files:
```

```
    - "file://state/backlog/b_2025-09-10.json"
    - "file://state/backlog/b_2025-09-11.json"
    - "file://state/backlog/b_2025-09-12.json"
runtime:
  freshness_requirement: "PT30S"
```

> 포인트: 결정적 정렬/윈도우/결합 규칙이 시나리오에 그대로 반영되므로, 재실행 시에도 동일한 입력 집합이 선별됩니다(환경이 같다면).

---

6) 2차 검증 — Schema/SHACL

- `requiredInputs` ⊆ `{job_route, machine_state, calendar, setup_matrix, wip_files, backlog_files}`
- 시간 필드가 ISO-8601인지, 경로/스킴이 정책에 맞는지, digest 패턴 등 확인.
- 실패 시 생성기는 **원인코드+메시지**로 Fail-Fast.

---

7) 매니페스트 — 재현성/추적성용 해시·출처 기록

```
{
  "goalId": "goal-job-eta-0001",
  "selectedModelRef": "aas://ModelCatalog/JobETAModel@1.4.2",
  "templateVersion": "job-eta@1.0.0",
  "inputs": {
    "JobRoute": { "resolved": "aas://FactoryTwin/JobRoute/JOB-7f2e3a8b-1
d" },
    "MachineState": { "resolved": "aas://FactoryTwin/State/Machine?at=202
5-09-12T15:13:31+09:00" },
    "Calendar": { "resolved": "file://config/factory_calendar.yaml", "digest":
"sha256:..." },
    "SetupMatrix": { "resolved": "file://config/setup_matrix.yaml", "digest":
"sha256:..." },
    "WIP": {
      "resolved": [
        "file://state/wip/wip_2025-09-12T15-10.json",
        "file://state/wip/wip_2025-09-12T15-05.json",
```

```
        "file://state/wip/wip_2025-09-12T15-00.json",
        "file://state/wip/wip_2025-09-12T14-55.json",
        "file://state/wip/wip_2025-09-12T14-50.json"
      ],
      "combine": { "op": "overlay", "key": "partId" },
      "digest": "sha256:..."
    },
    "Backlog": {
      "resolved": [
        "file://state/backlog/b_2025-09-10.json",
        "file://state/backlog/b_2025-09-11.json",
        "file://state/backlog/b_2025-09-12.json"
      ],
      "combine": { "op": "concat" },
      "digest": "sha256:..."
    }
  },
  "scenario": { "path": "./scenarios/goal-job-eta-0001.yaml", "digest": "sha2
56:..." },
  "frozenAt": "2025-09-12T15:13:31+09:00"
}
```


> manifest는 “무엇을 바탕으로 어떤 시나리오를 만들었는가”를 **내용주소(content-addressed)**로 고정합니다. 논문/감사에서 재현성 근거가 됩니다.

---

8) 실행·결과 매핑·AAS 업로드(개요)

1. 컨테이너 `registry/factory/job-eta:v1.4.2` 실행 → `scenario.yaml` 주입
2. 엔진 출력:


```
{ "completion_time": "2025-09-12T18:07:42+09:00",
  "tardiness_s": 1542.3,
  "sla_met": false }
```


3. 룰/매퍼가 `outputSpec`와 대조해 **AAS ExecutionStatus**로 업로드.

---

9) 한 줄 요약

- **입력은 형식화(QueryGoal+Meta+Bindings+Template)**,
- **시간은 실시간 치환(@현재시간 → ISO-8601)**,
- **선별·결합은 결정적 규칙(정렬/윈도우/overlay/concat)**,
- **산출물은 scenario.yaml + manifest.json**으로 **재현성 보장**.

원하시면 위 예시를 **그대로 실행 가능한 스켈레톤 코드(파이썬 CLI)**로 만들어 드릴게요.