SWRL 설계안: Goal2JobETA

0) 핵심 개체와 속성 (미니멀 온톨로지)

- **개체(Class)**
  - `ex:QueryGoal` / `ex:Model` / `ex:MetaDataFile`
  - `ex:RequiredInput` / `ex:ScenarioTemplate` / `ex:ScenarioPlan`
  - `ex:Execution` / `ex:OutputSpecItem` / `ex:UploadRequest`
- **주요 속성(Object/Data Properties)**
  - `ex:goalType(x, "predict_job_completion_time")`
  - `ex:hasParameter(x, "key", "value")` *(key–value를 개별 트리플로 전개)*
  - `ex:selectedModel(x, m)` *(참조 + 스냅샷은 별도 데이터 속성 보유)*
  - `ex:hasMetaData(m, f)` / `ex:hasRequiredInput(m, ri)`
  - `ex:providesInput(g, ri)` *(Goal이 해당 입력을 충족함)*
  - `ex:bindsTo(g, ri, "aas://...")` *(입력 바인딩 URI)*
  - `ex:hasScenarioTemplate(m, t)` / `ex:composedPlan(g, p)`
  - `ex:status(g, "Created|ModelResolved|Composed|ReadyToExecute|Executed|Uploaded")`
  - `ex:outputSpec(g, o)` / `ex:produced(g, "completion_time", ...)`
  - `ex:requestUpload(g)` *(업로드 트리거 신호)*

> 포인트: SWRL은 부정/집합/개체 생성에 제약이 있습니다.
>
> - *검증(누락 탐지·집합 포함)**은 **SHACL**로,
> - *개체 생성(ScenarioPlan 등)**은 **SWRLAPI의** `swrlx:makeOWLThing` 또는 **오케스트레이터에서 생성**으로 처리합니다.

1) 룰 세트 개요

- **R1. 모델 선택(Goal→Model)**
- **R2. 메타데이터 부착 확인(Model→MetaData)**

- **R3. 입력 충족/바인딩 맵 생성(Goal params → providesInput/bindsTo)**
- **R4. 협력 시나리오 구성(Templates → ScenarioPlan)**
- **R5. 실행 준비 상태 전이(Composed→ReadyToExecute)**
- **R6. 실행 완료 결과를 출력스펙에 매핑(Execution→produced)**
- **R7. 업로드 요청 신호 생성(UploadRequest)**

아래는 **SWRL 표기**로 제시하되, 엔진 의존 빌트인(예: `swrlx:makeOWLThing`, 사용자 정의 `ex:now`)은 코멘트로 표시합니다.

---

2) 룰 상세

R1) 모델 선택

> GoalType이 Job ETA이고, 카탈로그에 목적이 DeliveryPrediction인 모델이 있으면 선택.

```
ex:QueryGoal(?g) ^ ex:goalType(?g, "predict_job_completion_time") ^
ex:Model(?m) ^ ex:purpose(?m, "DeliveryPrediction") ^ ex:inCatalog(?m, tru
e)
→ ex:selectedModel(?g, ?m) ^ ex:status(?g, "ModelResolved")
```

- 구현 메모: `selectedModelRef` 문자열은 데이터 속성으로 병기하고, 삼중 저장소에는 `ex:selectedModel(?g, ?m)` 오브젝트 링크를 유지하는 것을 권장.

R2) 메타데이터 존재 확인

> 선택된 모델이 메타파일을 참조하면 OK 표식을 남김.

```
ex:selectedModel(?g, ?m) ^ ex:hasMetaData(?m, ?f)
→ ex:metaAttached(?g, ?f)
```

- 실패 케이스는 SHACL에서 `sh:Violation`으로 검출("메타 없음").

R3) 입력 충족/바인딩 생성 (파라미터→입력 매핑)

> Goal 파라미터를 모델의 requiredInputs로 끌어맞추는 매핑 룰들.

예) `jobId`가 있으면 `JobRoute`를 제공가능하다고 표시하고, AAS URI를 바인딩:

```
ex:QueryGoal(?g) ^ ex:hasParameter(?g, "jobId", ?jid)
→ ex:providesInput(?g, ex:JobRoute) ^
  ex:bindsTo(?g, ex:JobRoute, concat("aas://FactoryTwin/JobRoute/", ?jid))
```

실시간 스냅샷 입력은 "지금 시각"으로 바인딩:

```
ex:selectedModel(?g, ?m) ^ ex:hasRequiredInput(?m, ex:MachineState) ^
ex:QueryGoal(?g) ^ ex:hasParameter(?g, "bindAt", ?bindAt)     // 선택 파라미
터
→ ex:providesInput(?g, ex:MachineState) ^
  ex:bindsTo(?g, ex:MachineState, concat("aas://FactoryTwin/State/Machin
e?at=", ?bindAt))
```

> 현재시각 치환: @현재시간 탐지는 전처리 또는 사용자 정의 빌트인으로 처리
>
> *(예: `ex:now(?utc)`를 제공하고, `bindAt=@현재시간`이면 `?bindAt= ?utc`로 치환)*

- **전체 충족 판정**은 SHACL로 수행:

  `set(providesInput) ⊇ set(hasRequiredInput)` 이어야 통과.

R4) 협력 시나리오 구성 (템플릿 기반)

> 모델이 보유한 ScenarioTemplate들을 현재 Goal에 붙여 플랜으로 인스턴스화.

옵션 A — SWRLAPI 확장 사용(예: `swrlx:makeOWLThing`)

```
ex:selectedModel(?g, ?m) ^ ex:hasScenarioTemplate(?m, ?t) ^
ex:templateEnabled(?t, true) ^ ex:metaAttached(?g, ?f)
→ swrlx:makeOWLThing(?p) ^ rdf:type(?p, ex:ScenarioPlan) ^
  ex:planOf(?p, ?g) ^ ex:instantiatedFrom(?p, ?t)
```

옵션 B — 순수 SWRL + 오케스트레이터

- 룰은 **연결 사실만 생성**: `ex:eligibleTemplate(?g, ?t)`
- 오케스트레이터가 이 사실을 감시하여 `ScenarioPlan` 개체를 생성.

템플릿이 **다중 파일 입력**을 요구할 때, 템플릿에 `ex:combine`(union/concat/overlay/latest) 같은 데이터 속성을 미리 정의해 두고, 오케스트레이터가 `bindings.yaml`을 구성하도록 합니다.

R5) 실행 준비 상태 전이

> 모델 선택 + 메타 부착 + 입력충족(SHACL 통과) + 시나리오 구성 완료 → Ready.

```
ex:selectedModel(?g, ?m) ^ ex:metaAttached(?g, ?f) ^
ex:allInputsSatisfied(?g, true) ^ ex:hasPlanCount(?g, ?n) ^ swrlb:greaterTha
n(?n, 0)
→ ex:status(?g, "ReadyToExecute")
```

- `ex:allInputsSatisfied`와 `ex:hasPlanCount`는 **SHACL/오케스트레이터**가 계산하여 사실로 주입.

R6) 실행 결과의 출력 스펙 매핑

> 엔진이 결과를 저장하면, 출력스펙 이름과 타입에 맞춰 “produced” 사실을 생성.

```
ex:Execution(?e) ^ ex:ofGoal(?e, ?g) ^ ex:state(?e, "Completed") ^
ex:outputSpec(?g, ?o) ^ ex:specName(?o, "completion_time") ^ ex:resultVal
ue(?e, "completion_time", ?ct)
→ ex:produced(?g, "completion_time", ?ct)

ex:Execution(?e) ^ ex:ofGoal(?e, ?g) ^ ex:state(?e, "Completed") ^
ex:outputSpec(?g, ?o) ^ ex:specName(?o, "tardiness_s") ^ ex:resultValue(?
e, "tardiness_s", ?td)
→ ex:produced(?g, "tardiness_s", ?td)
```

- 결과-키(`"completion_time"`, ...)는 엔진 측이 `ex:resultValue`로 기록(키-값 저장소).
- 스펙과 일치하지 않는 키는 무시되거나 경고(선호: SHACL로 타입 검사).

R7) AAS 업로드 요청 트리거

> 필수 출력이 모두 생산되면 업로드 요청 사실을 생성 → 업로드 마이크로서비스가 처리.

```
ex:QueryGoal(?g) ^
ex:produced(?g, "completion_time", ?ct) ^
ex:produced(?g, "tardiness_s", ?td) ^
```

```
ex:produced(?g, "sla_met", ?sm)
→ ex:requestUpload(?g) ^ ex:status(?g, "Uploaded")
```

- 실제 AAS 업로드는 **부작용(side-effect)** 이므로, 마이크로서비스가 `ex:requestUpload(?g)` 를 구독하고 수행.

---

3) SHACL로 보강하는 핵심 검증(요지)

- **입력 충족**: `hasRequiredInput(model)` **모두**에 대해 `providesInput(goal)`이 있어야 함.
- **스펙 일치**: `produced(goal, name, value)`의 타입이 `outputSpec.datatype`와 합치.
- **메타·컨테이너 무결성**: `digest` 형식, `MetaData` 파일 해시 일치.

> SWRL은 "모두 만족(all)" 검사를 잘 못하므로, SWRL=추론, SHACL=검증을 병행하는 설계가 안전합니다.

---

4) 운영 시퀀스(요약)

1. **(입력)** `QueryGoal` 도착 → 파라미터 전개(`jobId` , `dueDate=@현재시간` 치환)
2. **R1–R2** 모델 선택 및 메타 확인
3. **R3** 파라미터→입력 바인딩 생성
4. **SHACL** 입력 충족 검사
5. **R4** 템플릿 기반 시나리오 플랜 구성 (다중 파일 결합 규칙 반영)
6. **R5** `ReadyToExecute` 전이 → 오케스트레이터 실행
7. **엔진** 실행/완료 → 결과 키-값 기록
8. **R6** 출력 스펙 매핑 → `produced` 사실 생성
9. **R7** 업로드 요청 → AAS 업로드 → `Uploaded`

---

5) 구현 메모 & 권장 값

- **룰 이름/버전**: `SWRL:Goal2JobETA@v1.0` (메이저 변경 시만 증가)
- **현재시각 빌트인**: `ex:now(?zdt)` 사용자 정의 또는 전처리
- **개체 생성**: 가능하면 SWRLAPI `swrlx:makeOWLThing` 사용, 불가시 오케스트레이터가 생성

- **다중 파일 결합**: 템플릿의 `combine`(union/concat/overlay/latest) 값을 **메타데이터/템플릿**에 내재화
- **상태값**은 **단일 값**만 유지(멀티 값 충돌 방지): 상태 전이는 오케스트레이터가 최종 커밋

---

6) 간단 예: 파라미터→입력 바인딩 룰 모듈

```
# JobRoute
ex:QueryGoal(?g) ^ ex:hasParameter(?g,"jobId",?jid)
→ ex:providesInput(?g, ex:JobRoute) ^
  ex:bindsTo(?g, ex:JobRoute, concat("aas://FactoryTwin/JobRoute/", ?jid))

# Calendar / SetupMatrix (정적 파일)
ex:selectedModel(?g, ?m) ^ ex:hasRequiredInput(?m, ex:Calendar)
→ ex:providesInput(?g, ex:Calendar) ^
  ex:bindsTo(?g, ex:Calendar, "file:///workspace/factory/config/factory_calendar.yaml")

ex:selectedModel(?g, ?m) ^ ex:hasRequiredInput(?m, ex:SetupMatrix)
→ ex:providesInput(?g, ex:SetupMatrix) ^
  ex:bindsTo(?g, ex:SetupMatrix, "file:///workspace/factory/config/setup_matrix.yaml")
```
