**SLICE (IoT 공통 SW 엔진)**

**개발 가이드**

**IoT연구본부**

2005-1-11.0

**개정 이력**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 버전 | 개정일자 | 개정내역 | 작성자 | 승인자 |
| 0.1 | 2018.06.15. | * 초안 작성 |  |  |
| 0.9 | 2018.09.10. | * 수정 및 보완 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목차**

[1. 개요 6](#_Toc524344547)

[1.1 목적 6](#_Toc524344548)

[1.2 범위 6](#_Toc524344549)

[1.3 용어 및 약어 6](#_Toc524344550)

[1.3.1 용어 6](#_Toc524344551)

[1.3.2 약어 6](#_Toc524344552)

[2. SLICE 서비스 플랫폼 7](#_Toc524344553)

[2.1 SLICE 서비스 플랫폼 구성 8](#_Toc524344554)

[2.2 SLICE 서비스 플랫폼 동작 모델 9](#_Toc524344555)

[3. 개발환경 구성 11](#_Toc524344556)

[3.1 설치화일 목록 11](#_Toc524344557)

[3.2 파일별 설치 가이드 및 주의 사항 11](#_Toc524344558)

[3.2.1 Java 8 설치 11](#_Toc524344559)

[3.2.2 Eclipse 설치 11](#_Toc524344560)

[3.2.3 Xtext 설치 12](#_Toc524344561)

[3.2.4 Lombok 설치 13](#_Toc524344562)

[3.2.5 SLICE 프레임워크 소스 가져와서 시작하기 13](#_Toc524344563)

[4. 서비스 시나리오 예제: 자동차 도메인에서 카시트 제어 14](#_Toc524344564)

[5. 지능명세 구성 15](#_Toc524344565)

[5.1 지능명세의 구성 및 예시 16](#_Toc524344566)

[6. 지능명세: 데이터 모델(Data Model) 17](#_Toc524344567)

[6.1 컨텍스트(Context) 18](#_Toc524344568)

[6.2 이벤트(Event) 및 예외처리(Exception) 19](#_Toc524344569)

[7. 지능명세: 제어 모델(Control Model) 20](#_Toc524344570)

[8. 지능명세: 에이전트 프로파일(Agent Profile) 22](#_Toc524344571)

[8.1 룰셋(Rule-set) 및 행위(Behavior) 22](#_Toc524344572)

[8.2 어댑터(Adaptor), 스트림(Stream), 래퍼(Wrapper) 23](#_Toc524344573)

[8.2.1 어댑터(Adaptor) 24](#_Toc524344574)

[8.2.2 스트림(Stream) 26](#_Toc524344575)

[8.2.3 래퍼(Wrapper) 26](#_Toc524344576)

[9. 디바이스 구현 27](#_Toc524344577)

[10. 배포 28](#_Toc524344578)

**그림 목차**

[그림 1 SLICE 서비스 플랫폼 구성도 8](#_Toc524344579)

[그림 2 SLICE 서비스 플랫폼 동작 모델도 9](#_Toc524344580)

[그림 3 Xtext 설치 12](#_Toc524344581)

[그림 4 SLICE 프레임워크 소스 가져오기 13](#_Toc524344582)

[그림 5 서비스 시나리오(카시트 제어) 구성도 15](#_Toc524344583)

[그림 6 지능명세 구성도 16](#_Toc524344584)

[그림 7 지능명세 작성 예 17](#_Toc524344585)

[그림 8 지능명세: 컨텍스트 및 이벤트 구성 예 18](#_Toc524344586)

[그림 9 지능명세: 컨텍스트 작성 예 19](#_Toc524344587)

[그림 10 지능명세: 이벤트 작성 예 20](#_Toc524344588)

[그림 11 지능명세: 제어 구성 예 21](#_Toc524344589)

[그림 12 지능명세: 제어 작성 예 21](#_Toc524344590)

[그림 13 지능명세: 에이전트 프로파일(룰셋 및 행위) 구성 예 22](#_Toc524344591)

[그림 14 지능명세: 에이전트 프로파일(룰셋 및 행위) 작성 예 23](#_Toc524344592)

[그림 15 지능명세: 에이전트 프로파일(어댑터/스트림/래퍼) 구성 예 24](#_Toc524344593)

[그림 16 지능명세: 에이전트 프로파일(어댑터) 작성 예(1) 25](#_Toc524344594)

[그림 17 지능명세: 에이전트 프로파일(어댑터) 작성 예(2) 25](#_Toc524344595)

[그림 18 지능명세: 에이전트 프로파일(스트림) 작성 예 26](#_Toc524344596)

[그림 19 지능명세: 에이전트 프로파일(래퍼) 작성 예 27](#_Toc524344597)

[그림 20 디바이스 구현 작성 예 28](#_Toc524344598)

[그림 21 배포 작성 예 29](#_Toc524344599)

# 개요

## 목적

본 문서는 외부환경에 설치된 다양한 센서장치를 통해 적응적으로 상황을 인식하고 액츄에이터 장치를 통해 상황에 맞는 액션을 실행하는 규칙 기반 시스템으로 정의되는 사물지능 공통 SW 엔진(SLICE 서비스 플랫폼)에 대한 개발 또는 이의 활용이 가능하도록 개발자 가이드 제공을 목적으로 한다.

## 범위

본 문서에서는 사물지능 공통 SW 엔진(SLICE 서비스 플랫폼) 구성 및 동작모델, 개발환경 그리고 예제 서비스 시나리오와 지능명세(데이터 모델, 제어 모델, 에이전트 프로파일)에 대해서 기술을 한다.

## 용어 및 약어

### 용어

|  |  |
| --- | --- |
| **SLICE 서비스 플랫폼** | 사물의 이동이 최소화된 환경에서 사물이 사용자의 행위와 주변 상황의 변화를 스스로 인지하여 실시간 판단과 대응을 수행하고, 피드백에 따른 학습을 통해 자율적으로 최적화해 가는 사물지능 공통 소프트웨어 |

### 약어

|  |  |
| --- | --- |
| **IoT** | Internet of Things |
| **SLICE** | Self-Learnable IoT Common sw Engine |
| **CEP** | Complex Event Processing |
| **KB** | Knowledge Base |
| **REST** | REpresentational State Transfer |
| **MQTT** | Message Queuing Telemetry Transport |

# SLICE 서비스 플랫폼

SLICE 서비스 플랫폼은 외부환경에 설치된 다양한 센서장치를 통해 적응적으로 상황을 인식하고 액츄에이터 장치를 통해 상황에 맞는 액션을 실행하는 규칙 기반 시스템으로 정의할 수 있다.

이를 구체화하면, 센서장치에 의해 획득된 외부환경 데이터는 SLICE 서비스 플랫폼에 정보(예: fact)로 저장되며, 이러한 정보를 저장/관리하는 모듈을 지식 베이스(KB: Knowledge Base)라 한다. 획득된 외부환경 데이터를 바탕으로 추론하여 상황을 인식하고 인식된 상황에 적절한 액션을 결정하는 과정은 규칙엔진(rule engine)에 의해 이루어지는데, 사전에 미리 명세된 다수 서비스 규칙의 추론을 통해 이루어진다.

규칙추론은 다수 서비스 규칙 간에 전제와 결론의 다단계 연쇄(chaining)을 통해 암묵적인 지식을 유도하거나 실행해야 하는 액션을 유도하는 과정이다. 유도된 새로운 암묵적인 상황 지식은 지식 베이스에 새로운 명시적 지식으로 저장되고, 유도된 액션은 액츄에이터 장치를 통해 실행된다. 따라서 서비스 규칙은 상황지식을 추론하는 상황규칙과 액션지식을 추론하는 액션규칙으로 구분할 수 있다.

상황규칙은 액션(예: THEN) 파트에 새로운 정보(또는 지식)을 지식 베이스에 저장하는 등 정보(예: fact) 자체를 유도/변경/생성하는 명령이 삽입된 서비스 규칙을 말한다. 액션규칙은 서비스 규칙의 액션(예: THEN) 파트에 외부환경 및 사용자와 상호작용을 담당하는 실행코드를 호출하는 명령이 삽입되어 있는 서비스 규칙을 말한다. 액션규칙은 조건(예: IF) 파트에 명세된 상황이 만족되면 해당 명령을 호출함으로써 응용이 외부환경과 상호작용하도록 한다.

여기에서 액션실행에 의해 외부환경에 변화가 생기면, 외부환경의 변화가 센서 장치에 의해 감지되고 감지된 외부환경 변화에 맞추어 새로운 액션이 유도되어 실행되기 때문에 SLICE 서비스 플랫폼을 중심으로 순환고리가 형성된다.

## SLICE 서비스 플랫폼 구성

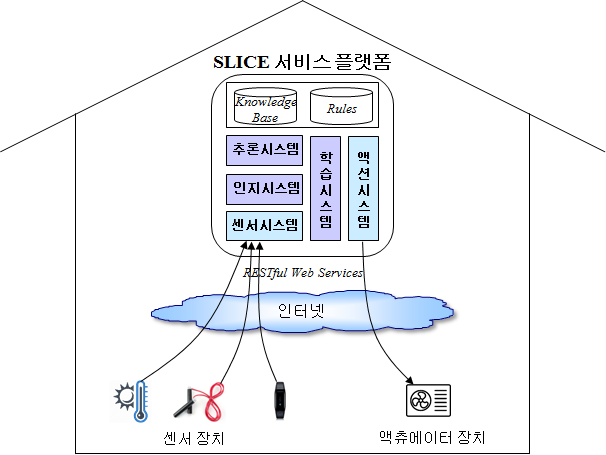


그림 1 SLICE 서비스 플랫폼 구성도

[그림 1]은 SLICE 서비스 플랫폼 구성을 도시하였다.

**센서 시스템**은 다양한 IoT 센서장치를 통해 외부환경으로부터 데이터를 획득하고 정제한다. 예를 들어 온도센서를 통해 실내온도(또는 습도센서를 통해 실내습도)를 일정 주기로 획득한 후, 데이터를 섭씨 온도로 변환하거나 오류 데이터 또는 잡음을 제거한다.

**인지 시스템**은 획득된 데이터를 단일 또는 다수의 IoT 서비스를 위해 미리 정의된 정보모델에 적합하도록 상황정보로 가공한다. 예를 들어 온도 센서를 통해 획득된 온도 데이터 29도를 ‘덥다’라는 상황정보로 가공한다.

**추론(의사결정) 시스템**은 상황정보를 저장하고 갱신/검색과 같이 상황정보를 관리하며 상황정보의 변경에 따라 다수 상황/액션 규칙을 추론하여 암묵적 지식과 상황변화에 대응하기 위해 필요한 액션을 유도한다. 예를 들어 실내 상황이 ‘덥다’이면 에어컨을 켜는 액션을 액션규칙 추론을 통해 유도한다.

**액션 시스템**은 다양한 액츄에이터 장치를 통해 액션을 실행한다. 예를 들어 에어컨을 켜는 액션이 유도되었을 때 실제 에어컨 장치를 제어해서 켜고 이를 통해 실제 외부환경에 변화를 가져온다.

**학습 시스템**은 사용자와의 상호작용을 지속적으로 관찰하여 액션규칙을 갱신한다. 예를 들어 사용자가 실내온도가 28도일 때 계속 에어컨을 켜는 것을 지속적으로 관찰하여 ‘실내 온도가 28도 이상이면 에이컨을 켜라’라는 액션규칙을 새롭게 추출한다. 기존 액션규칙을 갱신하거나 신규 액션규칙으로 추가한다.

## SLICE 서비스 플랫폼 동작 모델

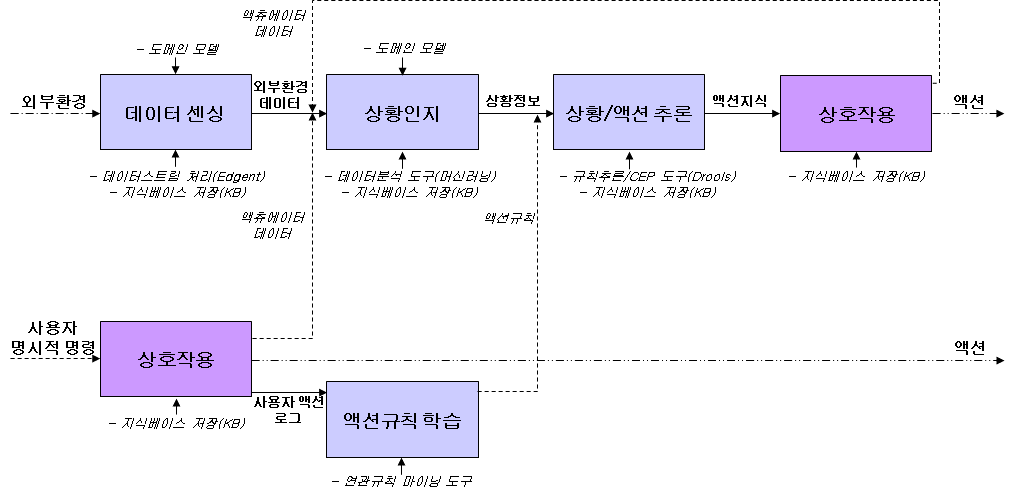


그림 2 SLICE 서비스 플랫폼 동작 모델도

[그림 2]는 SLICE 서비스 플랫폼 동작 모델을 도시하였다.

**“외부환경 데이터 센싱”** 행위(activity)를 통해 외부환경으로부터 다양한 도메인 모델에 관련된 데이터를 획득한다. 예를 들어 서비스 도메인이 댁내 환경이고 실내온도, 실내습도, 사용자 등이 도메인 모델로 정의되었다면, SLICE 서비스 플랫폼은 온도센서를 통해 실내온도를 획득하고, 습도센서를 통해 실내습도를 획득하고, 카메라를 통해 사용자 ID를 획득한다.

이때 획득한 데이터는 기본적으로 스트림으로 가정하고 이의 처리를 위해 SLICE 서비스 플랫폼은 SW 도구(예: Edgent)를 활용하여 획득주기를 조정하거나 필터링/병합 등의 기능을 제공한다. 또한 지식 베이스(KB: Knowledge Base)에 직접 저장될 수도 있고, 추가적인 정보 해석/추출을 위해 “상황인지” 행위(activity)의 입력 값으로 활용될 수 있다.

**“상황인지”** 행위(activity)에서는 데이터 센싱으로 획득된 외부환경 데이터를 도메인 모델에 따라 분석(해석 및 추출)한다. 예를 들어 실내온도 데이터는 도메인 모델에 따라 ‘적정’, ‘낮음’, ‘높음’ 등의 상황정보로 해석된다.

이때 데이터 분석 도구(예: 머신러닝)를 이용하여 구현된 영상분석 기능을 통해 영상 데이터로부터 위치(좌표) 등의 정보를 추출하는 과정을 수행할 수 있다. 따라서 “상황인지” 행위(activity)의 출력은 외부환경 데이터에 대한 해석 또는 새로운 정보가 추출되어 추상화된 상황정보가 되며, SLICE 서비스 플랫폼의 지식 베이스에 저장된다.

**“상황/액션 추론”** 행위(activity)에서는 지식 베이스에 저장된 상황정보를 활용하여 상황규칙의 추론 즉, 규칙 연쇄(chaning)를 통해 추가적인 또는 암묵적인 상황지식을 유도하여 지식 베이스에 저장하거나 액션규칙의 추론을 통해 현재 상황에서 외부환경 변화에 대응하여 실행할 액션지식을 유도한다.

**“상호작용”** 행위(activity)에서는 상황/액션 추론을 통해 유도된 액션을 외부환경 내 설치된 액츄에이터 장치를 기동시킴으로써 실행한다.

액션 실행 후에는 변화된 외부환경에 대한 데이터가 다시 센서 장치를 통해 획득되고, 이후 상기 절차에 의거하여 반복적으로 수행될 수 있다. 또한 액츄에이터 장치가 명령 실행 성공 후에 반환하는 값은 “상황인지” 행위(activity)의 입력으로 활용될 수도 있으며, 지식 베이스에 직접 저장될 수도 있다.

“상호작용” 행위(activity) 관련하여, 액션은 상황/액션 규칙 추론에 의해 유도되어 실행될 수도 있지만, 사용자의 명시적 명령에 의해 실행될 수도 있다. 또한 액션규칙 추론에 의한 액션 실행에서와 같이 액츄에이터 장치의 반환 값은 “상황인지” 행위(activity)의 입력으로 활용될 수도 있으며 지식 베이스에 직접 저장될 수도 있다.

실행된 사용자 액션의 내용은 이후 “액션규칙 학습” 행위(activity)을 위해 로깅되는데, 사용자 액션 로그에는 실행한 액션정보, 해당 액션실행 시점의 관련된 상황정보가 기록된다.

**“액션규칙 학습”** 행위(activity)에서는 사용자 액션 로그를 활용하여 새로운 액션규칙을 추출한다. 이때 연관규칙 마이닝 도구를 이용해 구현된 액션규칙 학습 기능이 활용된다. 추출된 액션규칙은 “상황/액션 추론” 행위(activity)의 입력 값으로 활용될 수 있다.

# 개발환경 구성

## 설치화일 목록

SLICE 서비스 플랫폼은 Eclipse를 기본 개발 환경으로 한다. SLICE 서비스 플랫폼을 개발하기 위해서는 아래와 같은 파일들의 설치가 필요하다.

* Java 8 설치: 기본 자바 툴 제공
* Eclipse 설치: 프로그래밍 통합 개발 환경 제공
* Xtext 설치: 지능 명세 변환 툴 제공
* Lombok 설치: 지능 명세로부터 생성된 코드 빌드에 활용
* GitHub으로부터 SLICE 프레임워크 소스 가져와서 시작하기

## 파일별 설치 가이드 및 주의 사항

### Java 8 설치

개발 환경 PC 의 코멘트 라인(Command Line)에서 “java –version” 을 수행하여 버전이 1.8 이상 설치 여부 및 “Java(TM) SE Runtime Environment” 가 표시되는지를 확인한다. 확인이 되지 않을 경우, 다음의 URL 에서 설치 파일을 받는다.

<https://www.oracle.com/technetwork/java/javase/downloads/index.html>

* Java SE Development Kit 로 설치할 것을 추천함

### Eclipse 설치

프로그래밍 통합 개발 환경인 Eclipse IDE 를 설치한다. 다음의 URL에서 설치 파일을 받는다.

<https://www.eclipse.org/downloads/>

2018년 9월 현재 Eclipse PHOTON 버전을 제공 중이며, 다운로드 후 시행 메뉴에서 Java 언어에 적합한 패키지를 선택하여 설치한다.

### Xtext 설치

Eclipse 개발 환경 내에서 추가로 필요한 툴들은 대부분 Eclipse 내의 “Elipse Marketplace” 메뉴에서 찾아서 설치가 가능하다.

SLICE 프레임워크에서 기술하는 지능 명세를 변환하기 위해서는 적절한 변환기가 필요하다. SLICE 프레임워크에서는 xText 를 사용한다.

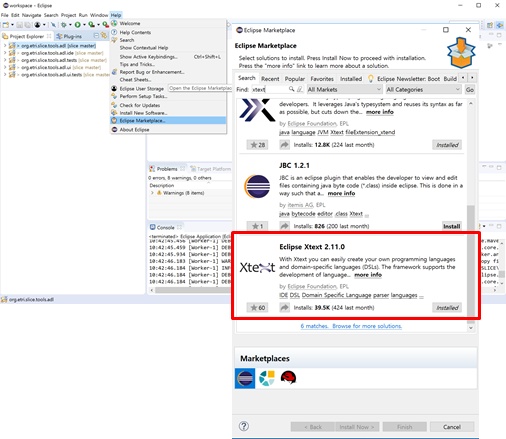


그림 3 Xtext 설치

상기 그림과 같이 Xtext 는 Eclipse 내의 “Elipse Marketplace” 메뉴에서 Search 기능을 통해 찾을 수 있으며, 이후 Install 버튼을 누르는 동작 만으로 간편하게 설치할 수 있다.

### Lombok 설치

보다 편리한 자바 코드 구현을 위해 SLICE 프레임워크에서는 Lombok을 사용한다. Lombok은 자바 클래스의 Getter/Setter 등을 자동으로 생성해 주는 툴이다. 소스 코드에 @Data, @@NoArgsConstructor, @AllArgsConstructor과 같은 annotation 들을 처리해 준다.

Lombok 툴은 아래의 URL에서 다운로드 받을 수 있다.

<https://projectlombok.org>

다운로드 후에는 임의의 설치 폴더에 복사한 후에, 설치된 폴더에서 코멘드 라인(Command Line)을 통해 “java –jar lombok.jar”을 실행한다.

### SLICE 프레임워크 소스 가져와서 시작하기

아래의 URL에서 SLICE 프레임워크 소스코드를 가져온다.

<https://github.com/slice-project/slice>

참고로 2018년 사물인터넷 국제전시회에 출품된 차량 시뮬레이터에 SLICE 프레임워크가 사용되었으며, 여기에 사용된 차량 응용 시나리오를 포함한 코드를 <https://github.com/slice-project/testbed>에 제공하고 있다. 이 코드에서는 여러 센서 장치에 대한 코드를 포함하고 있어서, SLICE 프레임워크를 이해하는데 도움이 된다.

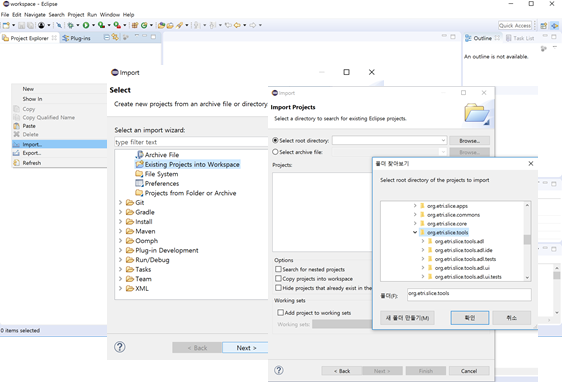


그림 4 SLICE 프레임워크 소스 가져오기

상기 그림과 같이, 가져온 SLICE 프레임워크 소스 코드를 적당한 폴더에 저장하고, Eclipse의 Import 기능 내의 “Existing Projects into Workspace”를 실행하여, Eclipse 내애서 작업할 프로젝트로 시작할 준비를 마친다

# 서비스 시나리오 예제: 자동차 도메인에서 카시트 제어

[그림 5]은 SLICE 서비스 플랫폼을 위한 서비스 예제로 자동차 도메인에서 카시트제어 시나리오를 도시한 것이다. 이는 임의의 사람이 차량에 접근하는지 센서를 이용하여 감시를 하고, 사람이 차량과의 일정거리에 들어오면 이벤트를 발생시켜서 사진을 찍는다. 이후 사람이 카시트에 앉으면 다양한 센서로부터 수집된 정보를 기반으로 카시트를 제어하는 것이다.

카시트 제어 서비스 시나리오를 지능명세의 관점에서 간략하게 기술하면 다음과 같다. 도메인 자동차내에서 다루어지는 데이터는 컨텍스트(Context)에서 정의를 하며, ObejectInfo, BodyPartLength, UserInfo, SeatPosture, Pressure가 컨텍스트에 해당된다. ObjectInfo는 Object가 감지되었을 때 사용되고, 감지하는 장치(예: 초음파)와 Object까지의 거리를 각각 objectId와 distance로 정의하고 있다. BodyPartLength는 사람 신체의 각 부분에 대한 길이를 정의하며, SeatPosture는 카시트의 상태를 나타내는 각도, 위치, 높이를 정의하고, Pressure는 압력센서에 가해진 압력값을 정의한다.

이벤트(Event)의 경우 ObjectDetected, FullBodyDetected, UserSeated, SeatPostureChanged 4가지를 정의하여 사용한다. Object가 감지되면 ObjectDetected 이벤트, 대상 Object의 길이가 산출되면 FullBodyDetected 이벤트, 사람이 카시트에 앉으면 UserSeated 이벤트, 카시트의 설정이 변경되면 SeatPostureChanged 이벤트가 각각 발생하도록 정의한다.

따라서, Object가 감지가 되면 ObjectInfo를 포함한 ObjectDetected 이벤트 발생, 대상 물체의 길이가 산출이 되면 대상 물체의 길이인 BodyPartLength를 포함한 FullBodyDetected 이벤트 발생, 사람이 카시트에 앉으면 UserInfo를 포함한 UserSeated 이벤트 발생, 카시트의 설정이 변경되면 변경된 카시트 설정값인 SeatPosture를 포함한 SeatPostureChanged 이벤트를 발생시켜 주고 받을 수 있도록 정의된 것이다.

제어 모델(Control Model)은 도메인 내에서 제어할 수 있는 내용에 대해서 정의하는 부분으로 예를 들어 Full-Body Detector는 임의의 조건 하에서 동작(예: 신체에 대한 사진을 찍고 각 부분의 길이 계산)을 정의하고, SeatPosture Controller는 카시트의 학습이나 제어에 의한 결과에 의해서 카시트의 상태를 변경하기 위한 동작을 정의한 것이다.

마지막으로, 임의의 조건에서 임의의 행동을 하도록 하는 룰 에이전트(Rule Agent)의 개념에서 조건과 행동을 가지는 룰셋(Rule-set)을 정의할 수 있으며, 이는 hasRuleSet과 hasBehaviors로 구성된다. 카시트 제어 시나리오 중에서 ObjectDetector 경우에 hasRuleSet 부분에서는 Objectdetector\_rules이라는 룰셋을 정의하는데, 이의 구성요소인 Group-id와 Artifact-id는 Objectdetector\_rules에 대한 검색이 가능하도록 하는 고유한 ID이다.

hasBehaviors 부분에서는 ObjectInfo가 발생되었을 때, ObjectDetected라는 이벤트를 발생시키도록 정의한다. 이는 사람이 다가와서 임의의 범위안으로 들어오면 Object가 감지되었다는 신호를 알려주기 위한 것이고, 그 정보는 ObjectInfo에 담아서 전달하게 된다.

각 기능의 구성 요소에 대한 상세한 설명은 다음 장에서부터 기술하겠다.

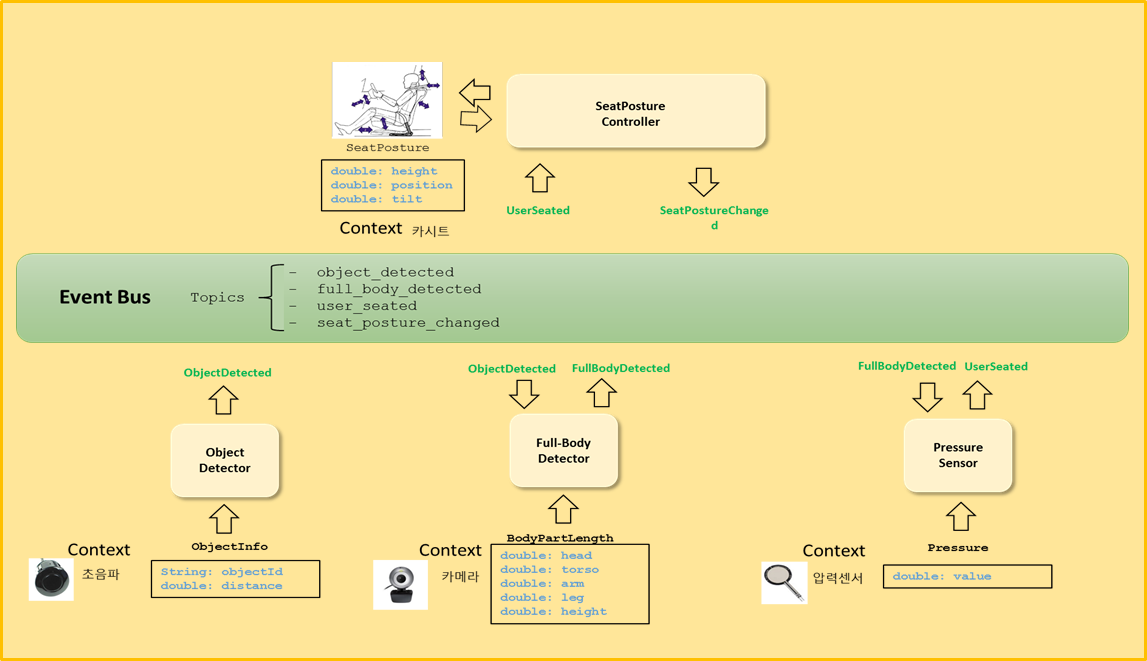


그림 5 서비스 시나리오(카시트 제어) 구성도

# 지능명세 구성

본 장에서는 지능명세의 구성에 대한 내용을 기술하였다. 지능명세는 크게 3부분으로 구성된다. 즉 데이터 모델(Data Model), 제어 모델(Control Model), 에이전트 프로파일(Agent Profile)이다. 데이터 모델은 컨텍스트(Context), 이벤트(Event) 및 예외처리(Exception)를 명세하며, 제어 모델은 제어(Control) 그리고 에이전트 프로파일은 룰셋(Rule-set) 및 행위(Behavior), 어댑터(Adaptor), 스티림(Steam), 래퍼(Wrapper)를 명세한다.

이의 세부적인 내용은 본 문서의 나머지 장에서 기술하였다. 또한 상기 내용에 대한 이해를 높이기 위해서 자동차 도메인에서 카시트 제어 시나리오를 기반으로 설명하였다.

## 지능명세의 구성 및 예시

[그림 6]은 자동차 도메인에서 카시트 제어를 위한 지능명세 작성시 요구되는 내용을 도시하였다. 본 문서에서 지능명세는 Agent Definition Language 확장자 파일(.adl )에 포함되어 있다.

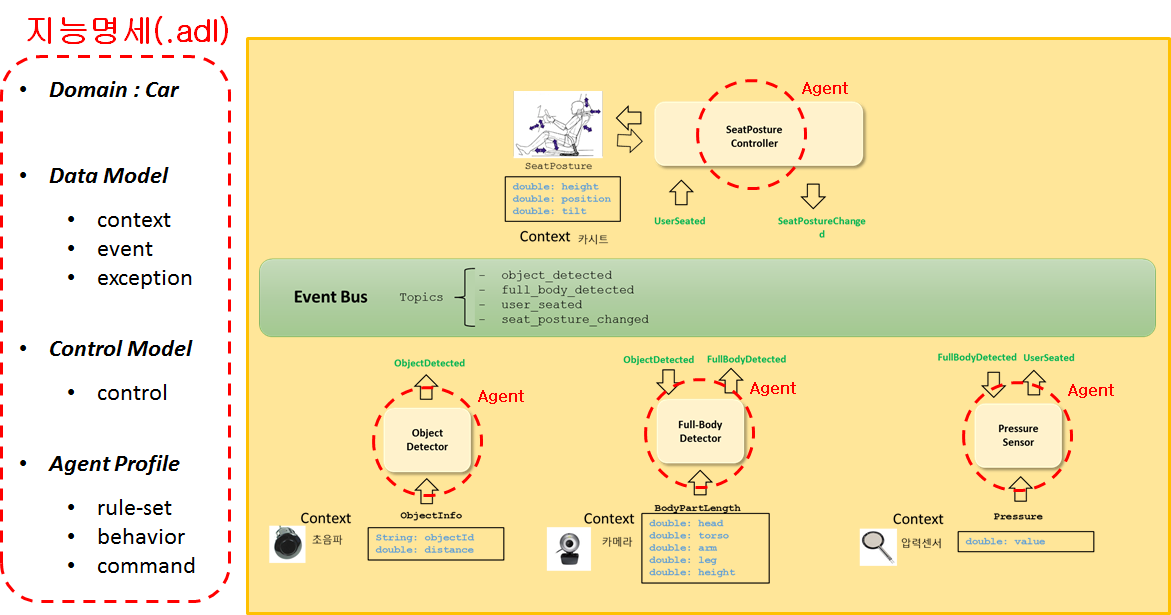


그림 6 지능명세 구성도

[그림 7]는 지능명세 작성을 도시하였다. 그림의 파랑색 점선의 내용과 같이 데이터 모델, 제어 모델, 에이전트 프로파일로 구성되는 carseat.adl 파일을 생성하면 지능명세 작성은 완료가 된다. 이후 실행을 위한 추가적인 로직 구성이 요구된다. 이를 위해 먼저 생성된 지능명세 파일을 빌드하면 SLICE 서비스 플랫폼은 상기 [그림 6]에 도시된 4개의 에이전트(즉 Object Detector, Full-body Detector, Pressure Sensor, SeatPosture Controller)를 기반으로 그림의 빨강색 실선에 포함된 패키지 및 파일들을 자동생성한다.

자동 생성된 패키지에서 Model 패키지(org.etri.slice.models)는 지능명세의 데이터 모델 관련 컨텍스트와 이벤트 및 제어 모델 관련 제어를 포함한다. Rule 패키지(org.etri.slice.rules)는 지능명세의 에이전트 프로파일 관련 룰셋을 포함하고, Agent 패키지(org.etri.slice.agents)는 에이전트 프로파일 관련 행위와 명령을 포함한다. Device 패키지(org.etri.slice.devices)는 디바이스 구현과 관련이 있고, Distribution 패키지(org.etri.slice.distribution)는 파일의 배포와 관련이 있다.

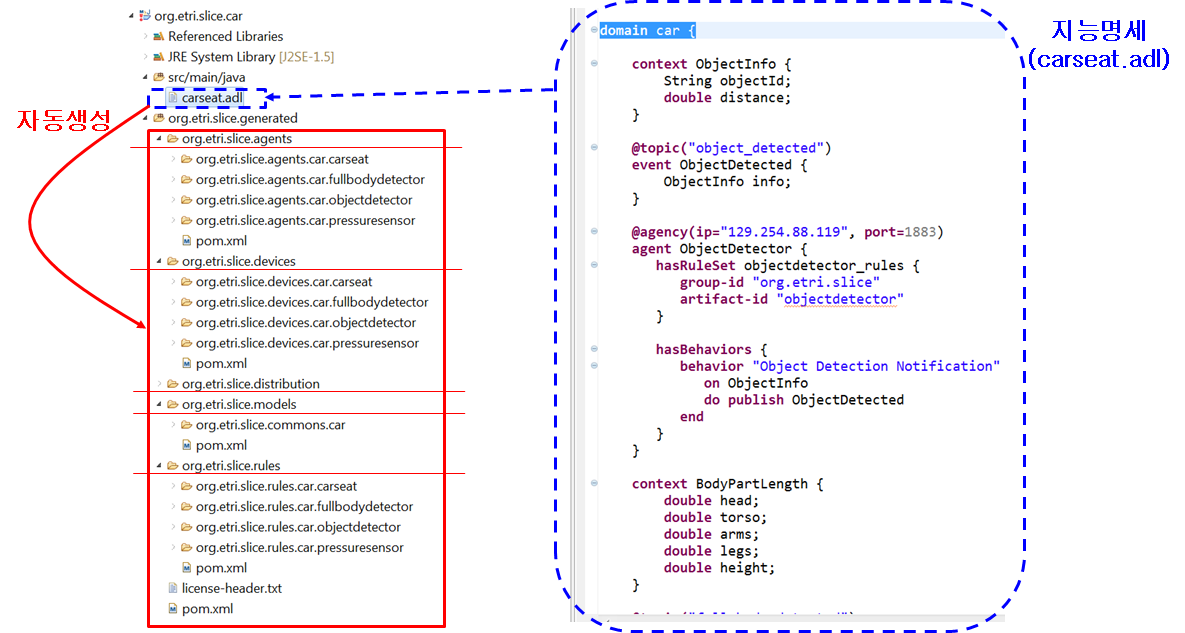


그림 7 지능명세 작성 예

# 지능명세: 데이터 모델(Data Model)

본 장에서는 지능명세의 데이터 모델에 대한 내용을 기술하였다. 데이터 모델은 컨텍스트(Context), 이벤트(Event), 예외처리(Exception) 명세를 포함한다. 컨텍스트는 센서 및 액추에이터로부터 발생하는 데이터의 타입 및 이름을 정의하는 것이고, 이벤트는 발생시 이벤트 버스(Event Bus)를 통해 공유되는 값들에 대해 정의하는 것이다. 예외처리는 예외적인 상황에 대한 처리를 기술하는 것이다. [그림 8]는 컨텍스트 및 이벤트 구성을 도시하였다.

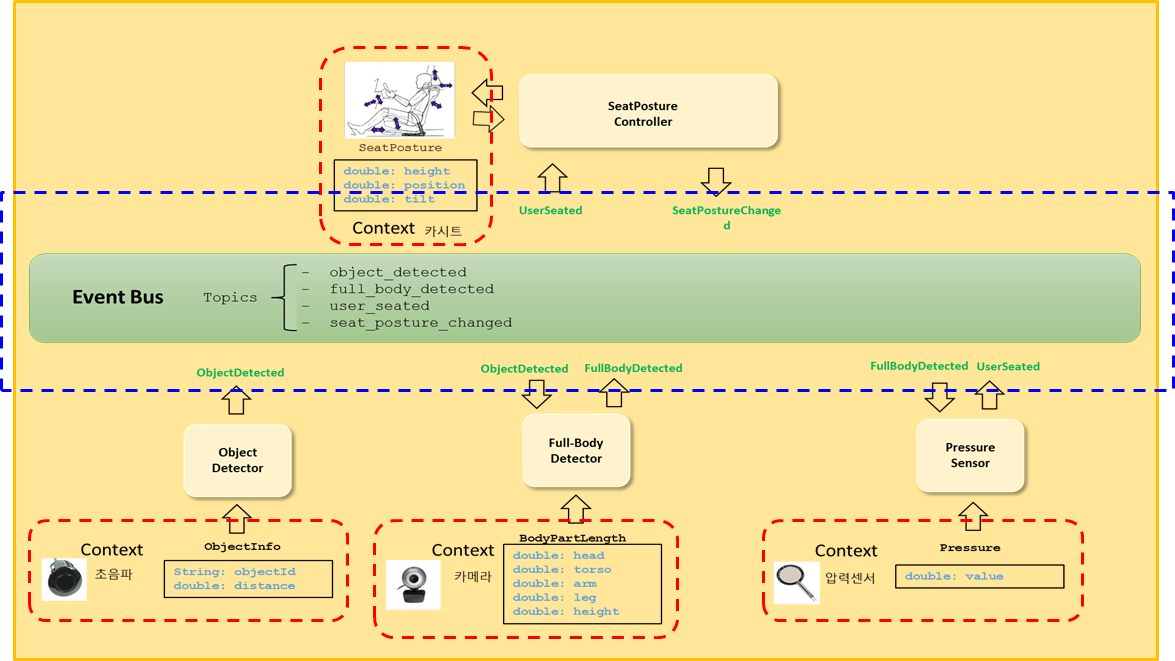


그림 8 지능명세: 컨텍스트 및 이벤트 구성 예

## 컨텍스트(Context)

자동차 도메인의 카시트 제어 시나리오에서 컨텍스트는 [그림 9]과 같이 ObejectInfo, BodyPartLength, Pressure, SeatPosture, UserInfo로 정의될 수 있다. ObjectInfo는 Object를 감지하기 위해서 사용하는데, Object까지의 거리를 포함한다. 또한 BodyPartLength는 사람 신체의 각 부분에 대한 길이 정보를 포함하며, Pressure는 압력센서에 가해진 압력값 그리고 SeatPosture는 카시트의 상태를 나타내는 등받이 각도, 카시트 위치 및 높이를 포함한다.

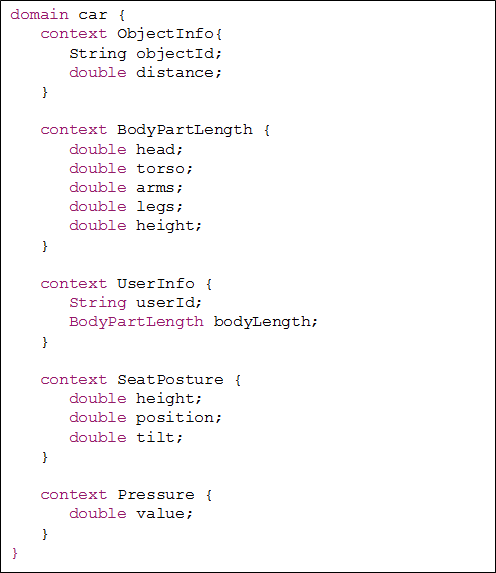


그림 9 지능명세: 컨텍스트 작성 예

## 이벤트(Event) 및 예외처리(Exception)

자동차 도메인의 카시트 제어 시나리오에서 이벤트는 [그림 10]과 같이 ObjectDetected, FullBodyDetected, UserSeated, SeatPostureChanged로 정의될 수 있다. ObjectDetected는 Object가 감지되었을 때, 컨텍스트에서 정의된 ObjectInfo 정보를 포함한 object\_detected를 발생하고, FullBodyDetected는 사람 신체의 각 부분에 대한 길이 정보가 산출이 되었을 때, BodyPartLength 정보를 포함한 full\_body\_detected를 발생한다. 또한 UserSeated는 사람이 카시트에 앉았을 때, UserInfo 정보를 포함한 user\_seated를 발생하며, SeatPostureChanged는 카시트의 설정이 변경되었을 때, SeatPosture 정보를 포함한 seat\_posture\_changed를 발생한다. 이때 도메인 내에서 이벤트 발생에 따른 값을 공유하기 위해 MQTT 프로토콜을 사용한다.

한편 예외상황에 대응하기 위해서는 [그림 10]의 이벤트와 동일한 방법으로 정의하여 처리를 할 수 있다.

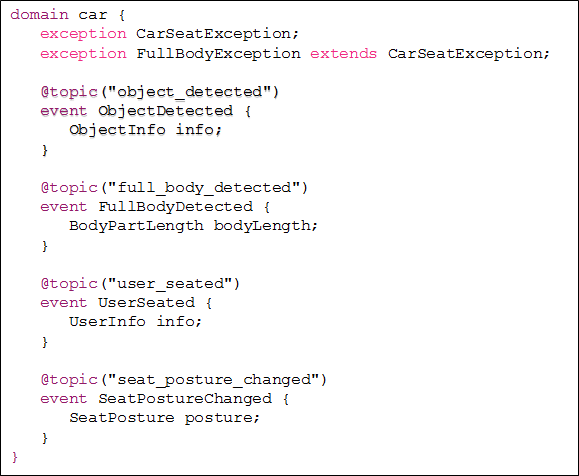


그림 10 지능명세: 이벤트 작성 예

# 지능명세: 제어 모델(Control Model)

본 장은 지능명세의 제어 모델에 대한 내용을 기술하였다. 제어 모델은 정의된 도메인 내에서 제어하고자 하는 내용을 미리 정의하는 부분으로 속성(Property)와 동작(Opeation) 두가지 형태로 가능하다. [그림 11]은 제어 구성을 도시하였다. Full-Body Detector의 경우 임의의 조건이 만족할 때 사진을 찍고 길이 분석 등과 같은 동작을 정의할 수 있으며, SeatPosture Controller의 경우 카시트의 학습 및 제어 결과에 의해서 카시트의 상태를 변경하기 위한 동작을 정의할 수 있다.

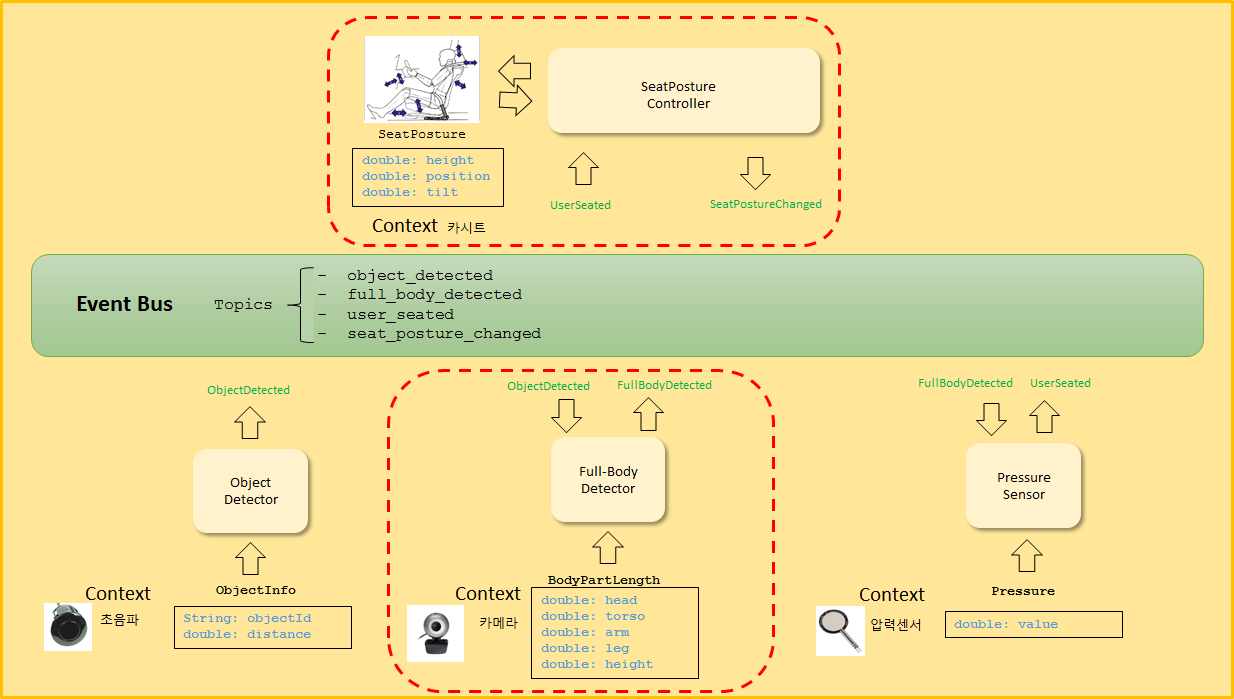


그림 11 지능명세: 제어 구성 예

자동차 도메인의 카시트 제어 시나리오에서 제어는 [그림 12]와 같이 정의될 수 있다. Startable은 자주 사용되는 Start와 Stop을 정의하기 위해 상용적으로 정의할 수 있는 부분이다. Full-Body Detector는 사진을 찍고 길이 분석 등을 하기 위한 동작을 정의하고 또한 Exception 처리를 할 수 있도록 정의하였다. SeatPosture Controller는 카시트의 상태를 변경하기 위한 속성을 정의할 수 있다.

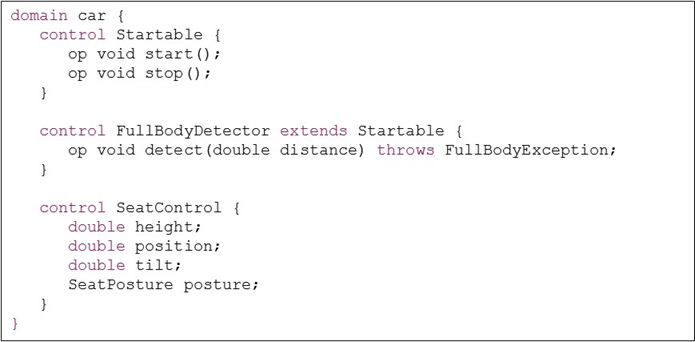


그림 12 지능명세: 제어 작성 예

# 지능명세: 에이전트 프로파일(Agent Profile)

본 장에서는 지능명세의 에이전트 프로파일에 대한 내용을 기술하였다. 에이전트 프로파일은 룰셋(Rule-set) 및 행위(Behavior), 어댑터(Adaptor), 스트림(Stream), 래퍼(Wrapper) 명세를 포함한다. 이의 세부적인 내용은 아래 절에서 기술하였다

## 룰셋(Rule-set) 및 행위(Behavior)

임의의 조건에서 임의의 액션을 하도록 하는 룰 기반의 에이전트의 개념에서 룰셋(Rule-set)은 ID를 정의하고 행위(Behavior)는 조건과 액션를 정의한다. [그림 13]은 룰셋 및 행위 구성을 도시하였다.

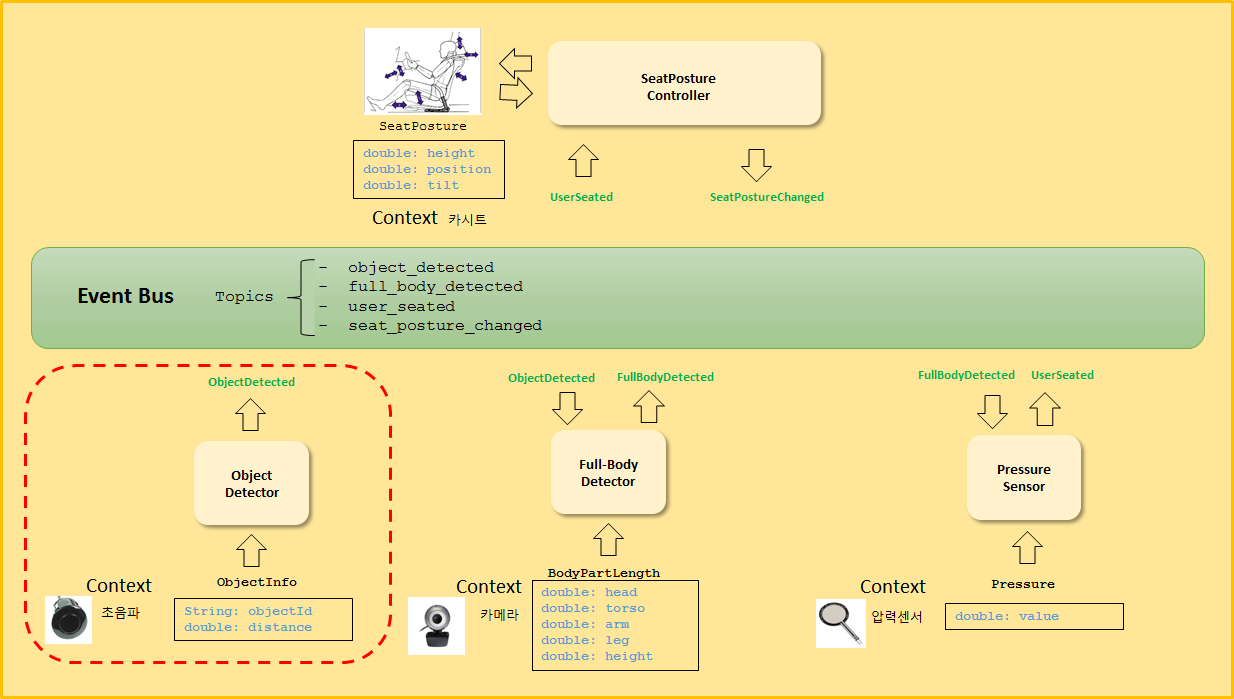


그림 13 지능명세: 에이전트 프로파일(룰셋 및 행위) 구성 예

룰셋(Rule-set) 및 행위(Behavior)는 자동차 도메인의 카시트 제어 시나리오에서 예를 들어 ObjectDetector의 경우 [그림 14]과 같이 작성될 수 있다. Group-id와 Artifact-id는 Objectdetector\_rules에 대한 검색이 가능하도록 하는 고유한 ID이다. 또한 ObjectInfo가 생성되었을 때 ObjectDetected라는 이벤트를 발생시키도록 ‘Object Detection Notification’이라는 Behavior를 정의하였다.

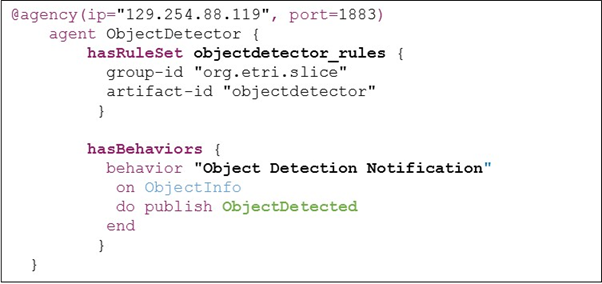


그림 14 지능명세: 에이전트 프로파일(룰셋 및 행위) 작성 예

## 어댑터(Adaptor), 스트림(Stream), 래퍼(Wrapper)

에이전트 프로파일에 기술되는 내용은 룰(Rule) 이외에 어댑터(Adaptor), 스트림(Stream), 래퍼(Wrapper)가 있다. [그림 15]는 어댑터/스트림/래퍼 구성을 도시하였다. SLICE 프레임워크에서 센서 또는 액츄에이터 및 이벤트 버스와 상호작용을 한다.

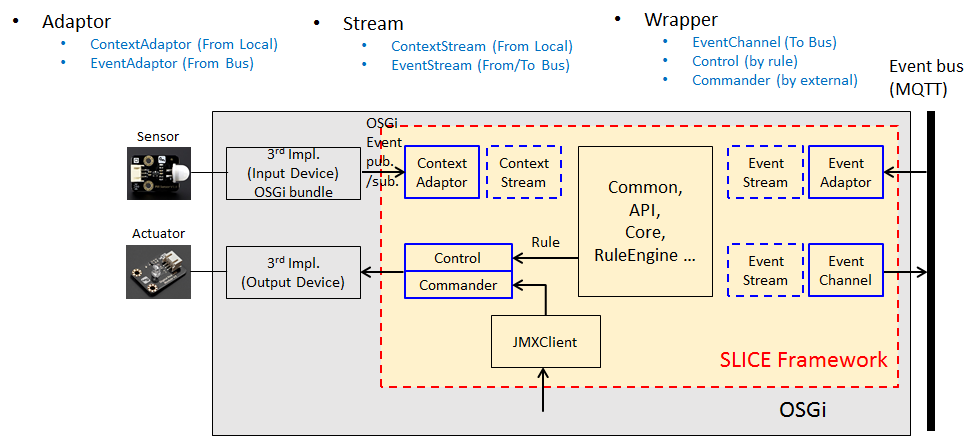


그림 15 지능명세: 에이전트 프로파일(어댑터/스트림/래퍼) 구성 예

### 어댑터(Adaptor)

어댑터(Adaptor)는 자동차 도메인의 카시트 제어 시나리오에서 예를 들어 ObjectDetector의 경우 [그림 16] 및 [그림 17]와 같이 작성될 수 있다.

SLICE 프레임워크는 어댑터 명세를 기술하여 외부 센서 또는 장치로부터 데이터를 OSGi 이벤트를 이용해서 전달받는다. 어댑터의 경우 에이전트의 행위(Behavior) 내에서 ContextName과 같이 기술된다. 기술된 명세에 의해 ContextNameAdaptor.java 파일이 생성된다. 예를 들어 ObjectDetector의 경우 ObjectDetector 에이전트 내에 ObjectInfo로 기술되어 있으면, ObjectInfoAdaptor.java 파일이 생성된다. ObjectInfoAdaptor는 EventSubsciber로부터 상속을 받으며, EventSubscriber는 OSGi에서 번들(Bundle)간 이벤트를 전달한다.

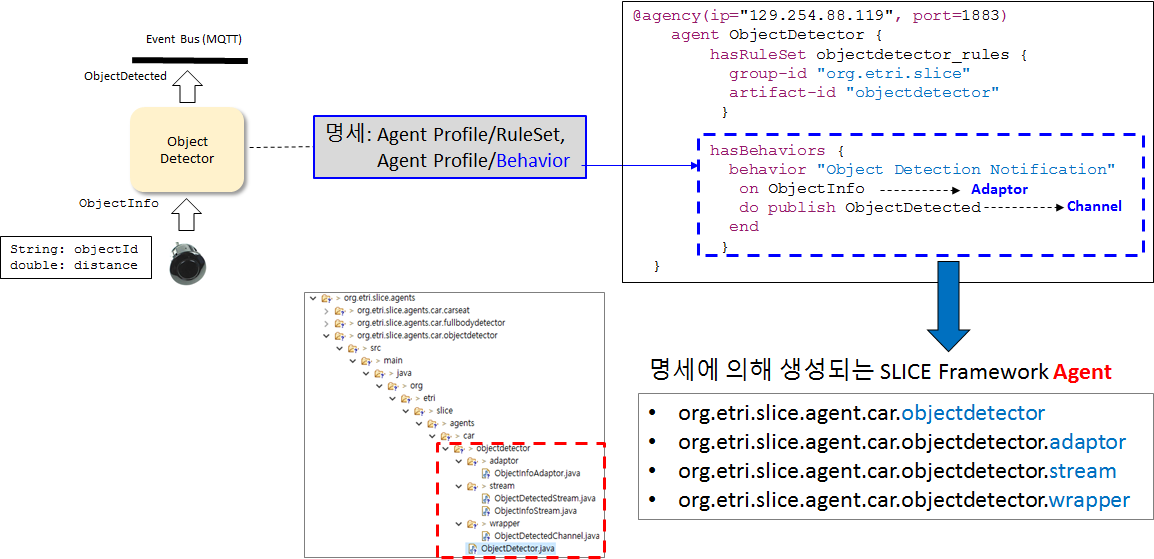


그림 16 지능명세: 에이전트 프로파일(어댑터) 작성 예(1)

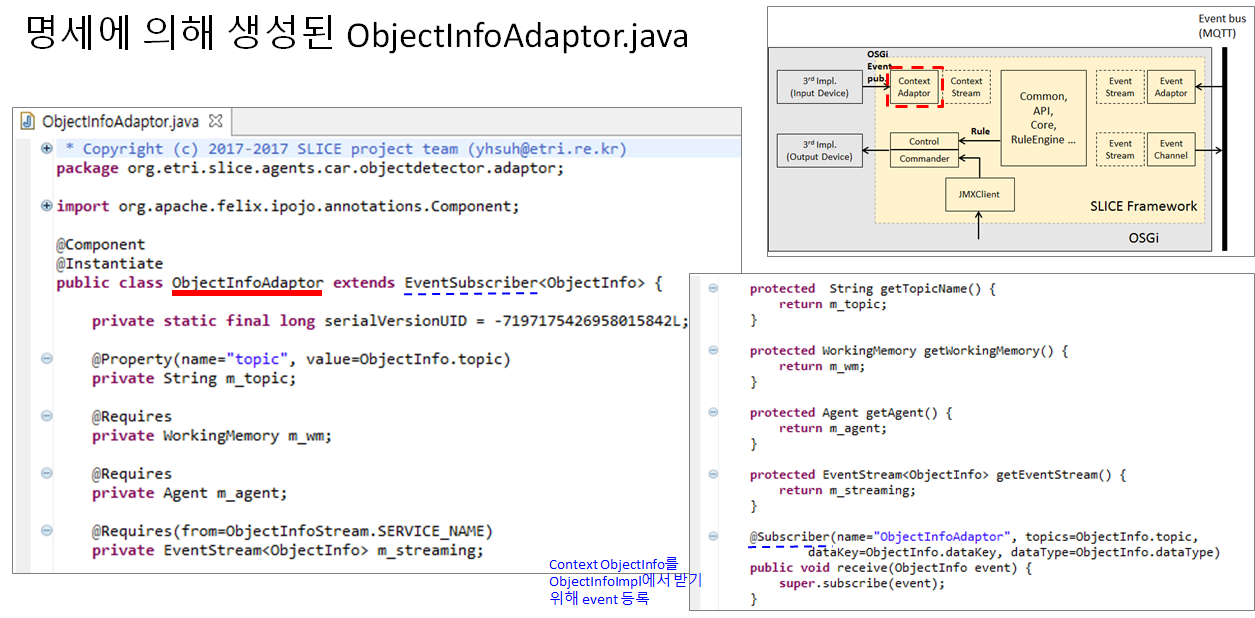


그림 17 지능명세: 에이전트 프로파일(어댑터) 작성 예(2)

### 스트림(Stream)

스트림(Stream)은 자동차 도메인의 카시트 제어 시나리오에서 예를 들어 ObjectDetector의 경우 [그림 18]와 같이 작성될 수 있다.

SLICE 프레임워크는 어댑터에서 전달된 데이터를 SLICE 코어에 적용이 용이하도록 데이터를 가공(예: 필터링)하는 기능을 수행하는 스트림 스텁(Stream Stub)을 자동으로 생성하여 제공한다. 이러한 데이터 가공 역할을 수행하기 위해 제공되는 스트림 스텁(Stream Stub)은 컨텍스트 어댑터(Context Adaptor), 이벤트 어댑터(Event Adaptor) 및 이벤트 채널(Event Channel) 각각의 스텁(Stub)들과 SLICE 프레임워크 코어 사이에 위치한다.

ObjectDetector 에이전트의 경우 기술된 명세에 따라 ObjectInfoStream.java와 ObjectDetectedStream.java를 생성 제공한다. ObjectInfoStream.java는 ObjectInfo 어댑터로부터 전달되는 데이터를 가공하여 SLICE 코어에 전달하고, ObjectDetectedStream.java는 SLICE 코어에서 전달된 데이터를 가공하여 ObjectDetectedChannel.java로 전달한다.

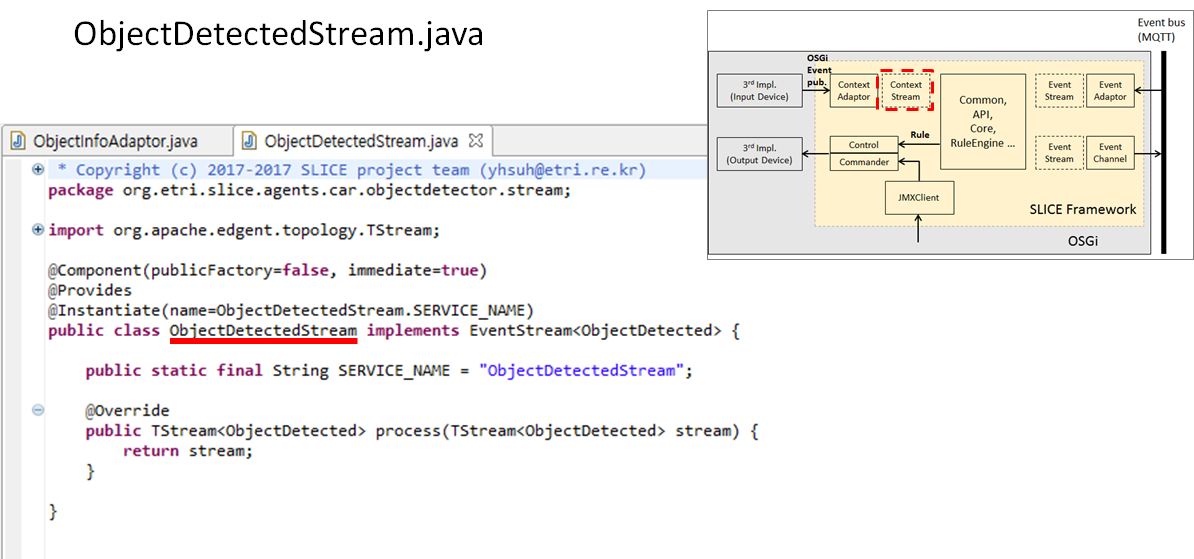


그림 18 지능명세: 에이전트 프로파일(스트림) 작성 예

### 래퍼(Wrapper)

래퍼(Wrapper)는 자동차 도메인의 카시트 제어 시나리오에서 예를 들어 ObjectDetector의 경우 [그림 19]과 같이 작성될 수 있다.

SLICE 프레임워크는 다른 지능형 IoT 디바이스에 데이터를 전달 또는 액츄에이터와 같은 디바이스를 제어하기 위해 래퍼 스텁(Wrapper Stub)을 자동으로 생성하여 제공한다. ObjectDetector 에이전트의 경우 기술된 명세에 따라 ObjectDetectedChannel.java를 생성하여 제공한다. ObjectDetectedChannel.java는 SLICE 코어에서 출력한 데이터를 이벤트 버스(MQTT)로 전달한다. ObjectDetectedChannel은 MqttEventPublisher로부터 상속받는다.

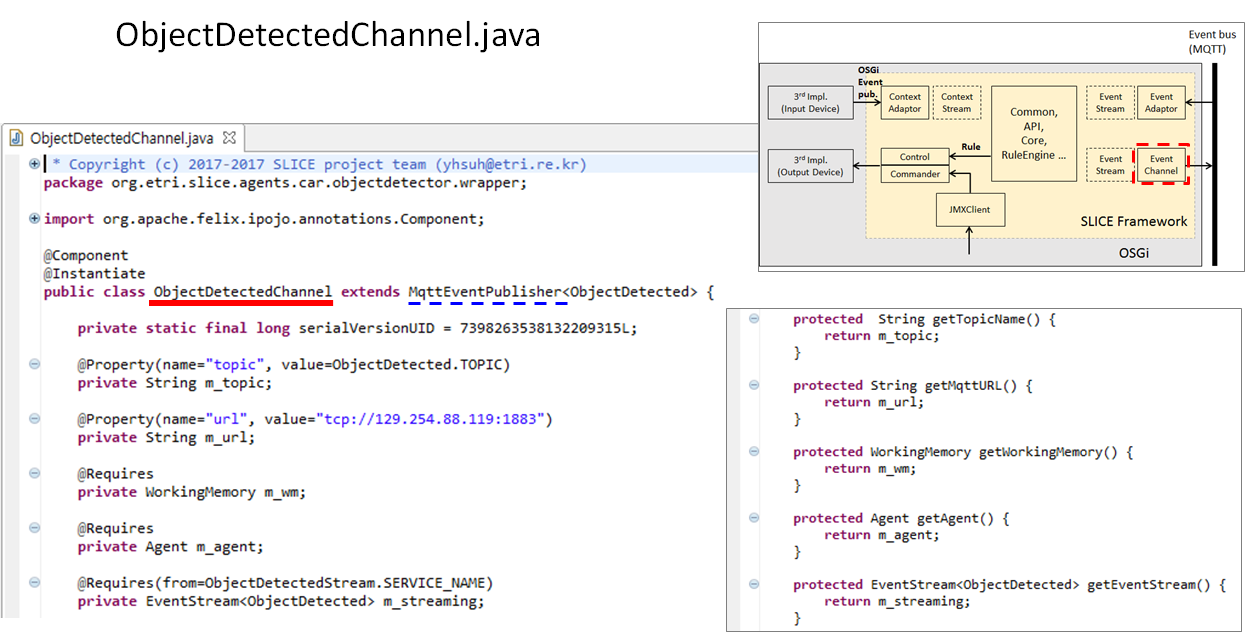


그림 19 지능명세: 에이전트 프로파일(래퍼) 작성 예

# 디바이스 구현

본 장에서는 디바이스 구현에 대한 내용을 기술하였다. 자동차 도메인의 카시트 제어 시나리오에서 예를 들어 ObjectDetector의 경우 디바이스 구현은 [그림 20]과 같이 작성될 수 있다. SLICE 프레임워크에서는 센서 및 액츄에이터 등의 디바이스를 제어하고 디바이스에서 제공하는 서비스를 사용하기 위한 클래스가 제공된다. 해당 클래스는 명세에서 컨텍스트 정의 및 제어에서 메소드가 정의되면 생성된다. 컨텍스트의 경우 생성된 클래스는 OSGi Event Publish를 이용하여 SLICE 프레임워크 어댑터로 데이터를 전달한다.

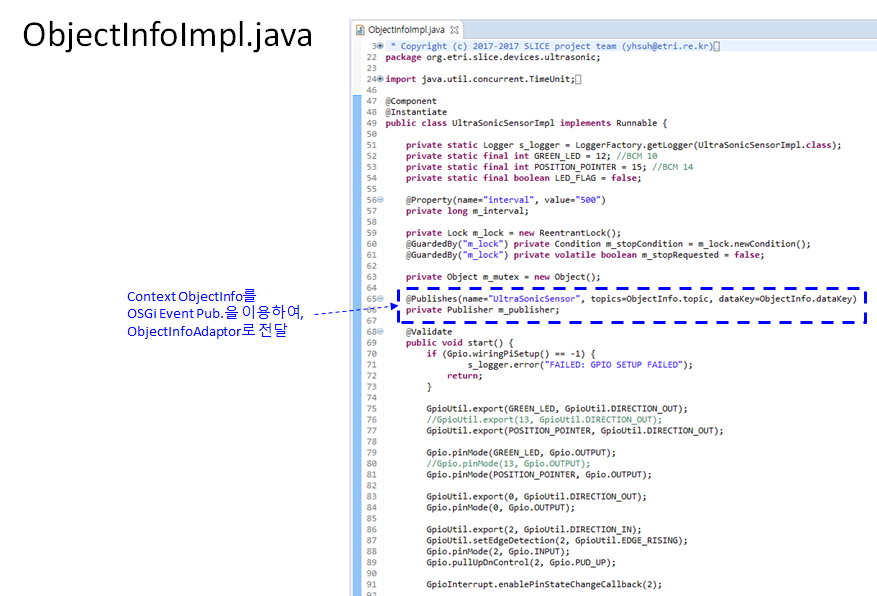


그림 20 디바이스 구현 작성 예

# 배포

본 장에서는 배포에 대한 내용을 기술하였다. 자동차 도메인의 카시트 제어 시나리오에서 예를 들어 ObjectDetector의 경우 배포는 [그림 21]과 같이 작성될 수 있다. 배포는 명세에서 정의된 각 에이전트를 독립된 자바 배포파일 .jar를 만들어준다. Maven의 pom.xml 파일을 수정하여 생성한다. ObjectDetector 에이전트의 경우 아래의 Dependency를 pom.xml에 포함하여 생성한다.

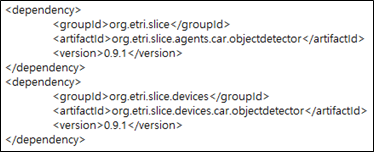


그림 21 배포 작성 예