한국정보전자통신기술학회논문지(jkiiect)'18-08, Vol.11 No.4
논문 18-11-04-379 http://dx.doi.org/10.17661/jkiiect.2018.11.4.379

지능형 IoE 플랫폼을 위한 Nools 기반 룰 엔진의 설계 및 구현

이성훈*, 황채은*, 정진우**

Design and Implementation of Nools-based Rule Engine for Smart IoE Platform

Sung-Hun Lee*, Chae-Eun Hwang*, and Jin-Woo Jeong**

요 약 IoT 생태계가 다양한 사용자들에 의하여 폭넓게 사용되기 위해서는 디바이스, 플랫폼과 사용자와의 상호작용 및 이에 대한 사용자 경험이 향상되어야 한다. 본 연구에서는 사용자들이 IoE (Internet of Everything) 플랫폼에서 디바이스와 서비스들을 보다 효과적으로 사용할 수 있도록 지원하기 위한 웹 기반 룰 엔진 시스템을 제안한다. 제안하는 시스템은 센서, 액추에이터, 외부 서비스 기반의 규칙 기반 시나리오를 직관적으로 생성하고 관리하기 위한 웹 인터페이스 및 UI 컴포넌트와 이를 통하여 생성된 규칙들을 Nools 룰 엔진에 적합하게 자동으로 변환하고 관리하기 위한 룰 관리 모듈로 구성되어 있다. 마지막으로, 본 논문에서는 각종 IoT 디바이스와 외부 서비스들의 룰 기반 응용 시나리오 동작 시연을 통하여 제안하는 시스템의 활용 가능성과 유용성을 보였다.

Abstract In order to make the IoT ecosystem more usable and friendly to general users, the interaction between the users and platforms/devices and its UX must be improved. In this study, a web-based rule engine system to help users intuitively interact with the various devices and services in IoE (Internet of Everything) platform is proposed. The proposed system consists of web interfaces and UI components for creating and managing rule-based scenarios using sensors, actuators, and external services in an intuitive way. Also, a rule management module for automatically translating the rules from the web interface into Nools rule engine compatible ones is proposed. Finally, we show the usability and feasibility of the proposed system by demonstration of rule-based use cases of IoT devices and external services.

Key Words : IoE, IoT, Nools, rule engine, web interface

1. 서론

최근, 스마트 디바이스 관련 기술의 발달에 따라 다양한 형태의 지능형 디바이스와 서비스의 상용화가 진행되고 있다. 기존의 사물 인터넷(Internet of Things: IoT)은 인터넷을 통하여 서로 연결되어 있는 디바이스 환경을 의미했다면, 최근에는 기존의 사물 인터넷 환경에서 확장된 만물 인터넷(Internet of Everything: IoE) 이라는 개념을 바탕으로 다양한 연구 개발이 진행되고 있다 [1]. 이러한 IoE 환경에서는 스마트 디바이스는 물론이고, 각종 센서, 빅데이터, 클라우드 등 디바이스 (사물) 뿐만 아니라 사용자, 프로세스, 데이터간의 연결성과 상호작용까지 가능한 모든 것을 연결하여 특정 목적의 기능들을 수행하게 된다. 이와 같이 융합 및 결합이 가능한 요소들의 형태와 수가 증가함에 따라 IoT(E) 플랫폼과 사용자 사이의 효과적인 상호작용을 위한 UX 연구의 중요성이 증가

This research was supported by The Leading Human Resource Training Program of Regional Neo Industry through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (2016H1D5A1910776).
* Department of Computer Engineering, Kumoh National Institute of Technology
** Corresponding Author: Department of Computer Engineering, Kumoh National Institute of Technology (jinw.jeong@kumoh.ac.kr)
Received July 07, 2018 Revised July 13, 2018 Accepted August 02, 2018

하고 있다. 이는 플랫폼에 등록되어 있는 디바이스의 조회, 검색과 같은 기본적인 시나리오뿐만 아니라 상황에 따라 디바이스, 서비스의 특정 기능을 수행하도록 하는 고급 시나리오를 위한 UX를 포함한다. 특히, 최근에는 디바이스나 사용자의 상황에 따라 적절한 액션을 수행하는 규칙 기반 시나리오를 지원하기 위한 다양한 형태의 시스템이 개발되고 있다.

IBM Watson IoT Platform [2], Thing+ [3], AWS IoT [4]등의 상용 시스템에서는 별도의 인터페이스를 제공하여 사용자가 디바이스의 상태에 따라 액션을 수행할 수 있는 규칙 생성 및 관리 기능을 이용하도록 하고 있다. 그림 1은 AWS IoT 플랫폼의 규칙 생성 관련 인터페이스를 나타내고 있다. 그림 1에서 볼 수 있는 것처럼, 상용 IoT 플랫폼들이 디바이스의 상황에 따라 액션을 수행하는 인터페이스를 제공하기는 하지만 그림 1의 “rule query statement”, “attribute”, “topic filter”와 같이 일반적인 스마트 홈 사용자들이 활용하기에는 복잡하고 어려운 사용법을 숙지해야 한다는 한계점이 존재한다. 이러한 형태의 인터페이스는 규칙 기반 시나리오의 관리 및 사용에 대한 사용자 편의성을 심각하게 저하시킬 수 있다.

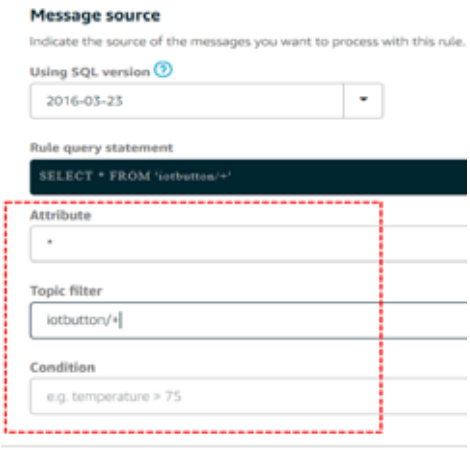

그림 1. AWS IoT 룰 엔진
Fig. 1. AWS IoT Rule Engine

상용 시스템뿐만 아니라 [5 – 11]과 같은 연구에서도 규칙 기반의 IoT 응용 시나리오 지원을 위한 다양한 접근법들을 제안하였다. [5]의 연구는 ECA (Event-Condition-Action) 규칙 기반의 제조 기업 현장관리 서비스 IoT 아키텍처를 제안하였고, [6]에서는 IFTTT 서비스를 위하여 CEP 엔진 기반의 실시간 이벤트 처리 및 규칙 관리 시스템을 제안하였다. 또한, [7]의 연구를 통하여 IoT 디바이스의 확장성을 고려한 룰 기반 자동 제어 관리 프레임워크가 제안되었으며 [8]의 연구는 확장 가능한 IoT 통합 관리 및 규칙 기반 자동화 시스템을 제안하였다. 한편, [9, 10, 11]등의 연구에서는 규칙/패턴 인식 알고리즘들을 활용한 상황 인지 기반 서비스 개발이 진행되었다. 그러나 이와 같은 연구들은 규칙 기반 IoT 서비스를 제공하기 위한 룰 엔진 시스템, 디바이스 및 데이터 구조에 집중하고 있어 사용자 편의성과 관련된 부분에는 한계점이 여전히 존재한다. 또한, 디바이스 상태 변화에 따른 액션 수행과 같이 디바이스 위주의 단순 조건문 생성에 초점을 맞추고 있으며, 연산자 종류가 다양하지 않아 룰을 자유롭게 생성하는데 한계가 있다.

본 논문에서는 보다 직관적이고 지능적인 IoE 상호작용을 지원하기 위하여 IoE 플랫폼을 위한 룰 엔진을 구축하고 이에 기반을 둔 웹 인터페이스를 제안한다. 이를 위하여 본 연구에서는 자체 제작한 IoE 플랫폼 및 프로토콜을 바탕으로 디바이스, 사용자, IoT 서비스간의 규칙 기반 응용 시나리오를 지원할 수 있는 웹 인터페이스를 구현하며, 오픈소스 프로젝트 Nools 룰 엔진을 활용하여 지능형 IoE 플랫폼을 위한 룰 엔진의 설계 및 구현을 수행한다. 이를 통한 본 연구의 주요 기여는 다음과 같다. 첫째, Node.JS 기반의 범용 비즈니스 룰엔진인 Nools 엔진을 수정 및 개선하여 IoE 플랫폼에 통합함으로써 지능형 시나리오 실현의 인프라 구축을 완료하였다. 둘째, 일반적인 센서뿐만 아니라 액추에이터, OpenAPI와 같은 외부 서비스를 이용한 지능형 시나리오를 수행할 수 있도록 IoE 플랫폼과 룰엔진 구조를 개선하여 보다 다양한 형태의 서비스를 생성 및 관리할 수 있도록 하였다. 셋째, 디바이스나 서비스별 웹 인터페이스와 UI 컴포넌트를 개발함으로써 사용자들이 복잡한 룰 언어 문법을 모르더라도 UI를 이용하여 직관적으로 관리가 가능하도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안하는 시스템에 대하여 기술하며 3장에서는 제안하는 시스템의 구현 내용 및 응용 시나리오에 대하 서술한다. 마지막으로 4장에서 결론과 향후 연구에 대하여 기술한다.

2. 제안하는 시스템

2.1 Nools: Javascript-based Rule Engine

Nools [12]는 javascript 언어로 작성된 RETE 기반의 오픈소스 룰 엔진 프로젝트로서 패턴 매칭과 규칙 기반 시나리오가 필요한 다양한 도메인에서 사용될 수 있다.

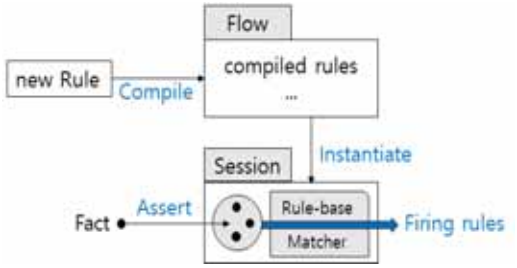

그림 2. Nools 룰 엔진
Fig. 2. Nools Rule Engine

그림 2는 Nools 룰 엔진의 시스템 구조를 나타내고 있다. Nools 룰 엔진은 크게 rule, fact, flow, session으로 구분되는 4개의 컴포넌트로 구성되어 있다. Rule은 규칙을 의미하며 규칙이 수행 될 조건을 나타내는 when절과 수행되는 동작을 나타내는 then을 정의할 수 있다. Fact는 일반적인 데이터를 의미하며 각각의 도메인마다 필요한 형태로 데이터의 구조를 정의할 수 있다. 또한, rule의 when절에서는 주로 fact의 상태나 속성 정보를 체크함으로써 수행 조건을 구성하게 된다. Flow는 rule들의 저장소라고 할 수 있으며 flow를 instance화 되면 session을 획득할 수 있다. Session은 rule들이 저장되어 있는 flow의 인스턴스로서 fact의 삽입, 수정, 삭제가 발생하는 메모리를 유지하며 저장되어 있는 fact와 rule을 바탕으로 패턴 매칭 및 그에 따른 rule의 실행이 발생한다.

Nools 엔진의 규칙(rule)은 '조건'-'동작'의 형태로 기술되며 프로그래밍 언어 혹은 Nools DSL (Domain Specific Language)를 이용하여 정의할 수 있다. 표 1과 2는 Session에 "hello world"와 정확히 일치하는 텍스트 속성을 가지는 메시지 객체가 존재할 경우 해당 메

표 1. 룰 정의 (프로그래밍 기반)
Table 1. Defining a rule (programmatically)

```
var Message = function (message) {
  this.text = message;
};

var flow = nools.flow("Hello World", function (flow) {
  flow.rule("Hello", [Message, "m", "m.text =~ /^hello
\\sworld$/"], function (facts) {
      facts.m.text = facts.m.text + " goodbye";
      console.log(facts.m.text);
      this.modify(facts.m);
  });
});
```

표 2. 룰 정의 (DSL)
Table 2. Defining a rule (DSL)

```
define Message {
  text : '',
  constructor : function(message){
      this.text = message;
  }
}

rule Hello {
  when {
      m : Message m.text =~ /^hello\\sworld$/;
  }
  then {
      modify(m, function(){this.text += " goodbye";});
      console.log(m.text);
  }
}
```

표 3. Fact의 삽입 및 수정
Table 3. Asserting and modifying facts

```
session.assert(new Message("hello"));
session.assert(new Message("hello world"));
session.assert(new Message("goodbye"));

var m = new Message("hello");
session.assert(m);

m.text = "hello goodbye";
session.modify(m);
```

시지 객체의 텍스트에 “goodbye”라는 텍스트를 추가한 후 콘솔에 해당 내용을 출력하는 규칙을 나타내고 있다. 표 1과 2는 각각 javascript와 Nools DSL을 이용하여 위의 rule을 정의하는 방법을 나타낸다. Javascript를 이용할 경우 flow 객체의 rule 메소드에 조건문을 설정하고 해당 메소드의 콜백 함수에 동작에 해당하는 구문을 정의한다. DSL을 이용할 경우 보다 직관적으로 각 규칙의 when/then 절을 정의할 수 있다. 표 1과 2의 Hello rule의 조건 절 검사에 필요한 fact의 구조는 Message 함수(객체)로 정의되어 있는 것을 확인할 수 있다. Nools 엔진의 fact는 문자열, 숫자, 날짜, 객체 등 다양한 형태의 타입을 지원하며, session에 fact를 추가/수정/삭제하고자 할 경우에는 표 3과 같이 처리할 수 있다.

2.2 Nools 기반 IoE 룰 엔진

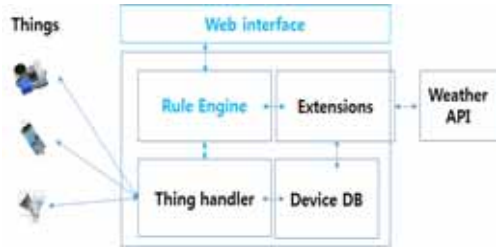

그림 3. IoE 플랫폼 구조
Fig. 3. IoE platform architecture

본 연구에서는 지능형 IoE 관리 및 상호작용을 위하여 자체 구축한 IoE 플랫폼에 Nools 기반의 룰 엔진을 적용하여 웹 인터페이스를 기반으로 사물과 서비스, 사용자간의 지능형 시나리오를 지원한다. 그림 3은 자체적으로 구축한 IoE 플랫폼의 구조를 나타내고 있다. IoE 플랫폼은 플랫폼 자체적으로 지원하는 사물들 (센서 및 액추에이터)의 연결 및 등록, 통신을 위한 프로토콜을 제공하며, 외부 서비스 (예: 날씨 API 등)를 플랫폼에 연동하여 사용할 수 있도록 확장 모듈을 제공한다. 본 논문에서는 플랫폼에 연동되어 서비스 중인 사물들과 외부 서비스를 바탕으로 사용자가 손쉽게 지능형 시나리오를 생성, 관리할 수 있도록 Nools 기반의 룰 엔진 구성과 웹 인터페이스 구축을 수행한다.

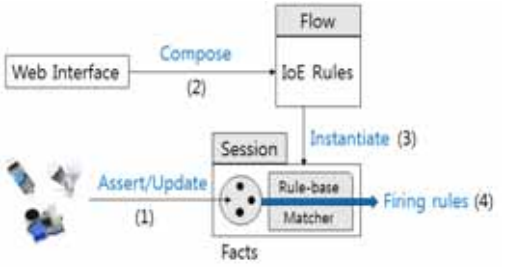

그림 4. IoE 규칙 기반 시나리오 작업 흐름
Fig. 4. IoE rule-based scenario workflow

그림 4는 제안하는 시스템의 규칙 기반 지능형 시나리오의 작업 흐름을 나타내고 있다. 먼저, 플랫폼의 디바이스 연결 및 등록 프로토콜을 바탕으로 플랫폼 디바이스 등록과 rule 및 fact-base의 초기화를 수행한다 (과정 1). 디바이스 혹은 외부 서비스가 플랫폼에 정상적으로 등록된 후에는 웹 인터페이스를 통하여 지능형 시나리오를 위한 규칙을 작성/수정한다 (과정 2). 웹 인터페이스를 통하여 작성이 완료된 규칙은 Nools DSL 형태로 변환되어 session에 추가된다 (과정 3). 그 후, 주기적으로 사물 혹은 외부 서비스의 상태가 변경될 때마다 플랫폼에 상태 갱신 정보가 플랫폼의 Things handler (그림 3의 Things handler)를 통하여 전송되며 Rule engine과의 연동을 통하여 fact-base에 정보를 갱신한다. 각각의 사물 및 서비스들의 상태가 사용자가 정의한 규칙에 일치하는 상황이 발생하는 경우 저장되어 있는 액션이 수행되어 규칙 기반의 시나리오를 제공한다 (과정 4).

표 4는 플랫폼의 디바이스와 외부 서비스 (날씨 정보), 시간 정보의 추가와 갱신에 필요한 fact message 구조의 정의를 나타낸다. Things Fact와 Weather Fact는 Nools DSL을 이용하여 사물과 날씨 정보를 룰 엔진의 fact-base에 추가하기 위하여 정의한 구조로 새로운 디바이스가 추가되거나 디바이스/날씨 정보가 갱신되었을 때 해당 구조를 이용하여 정보를 추가/갱신한다. Clock 구조는 “3분 뒤에 불 꺼줘”와 같이 시간과 관련된 규칙 기반 시나리오 지원을 위해 필요한 구조로 Javascript를 이용하여 정의되었으며 시/분/초 데이터의 획득뿐만 아니라 알람 (시각 확인), 기간 확인 등의 기능을 제공한다.

표 4. IoE 룰 엔진을 위한 Fact 정의
Table 4. Defining facts for IoE rule engine

```
define ThingsFact {
  name : 'thing',
  p1: '',
  p2: "",
  ...
  constructor : function(message){
    this.p1 = message.p1;
    this.p2 = message.p2;
    ...
  }
}

define WeatherFact {
  name : 'weather',
  weather_today: '',
  temp_today: "",
  ...
  constructor : function(message){
    this.weather_today = message.weather_today;
    this.temp_tdaoy = message.temperature_today;
    ...
  }
}

var Clock = function(){
  this.name='clock';
  this.date = new Date();
  this.getHours = ()=> {return this.date.getHours();};
  this.getMinutes= ()=> {return this.date.getMinutes();};
  ...
  this.isMinute= (min) => {
    return this.getMinutes() == min;
  };
  ..
  this.hoursIsBetween = (a, b) => {
    return this.date.getHours() >= a &&
           this.date.getHours() <=b;
  };
  this.timeComp= (dat) => {
    dat = new Date(dat);
    return this.isHour(dat.getHours()) &&
           this.isMinute(dat.getMinutes()) &&
           this.isSeconds(dat.getSeconds());
  };
}
```

사용자는 웹 인터페이스를 통하여 사물, 서비스를 대상으로 하는 규칙 기반 시나리오를 생성 및 관리할 수 있다. 인터페이스에서는 규칙의 이름 및 간단한 설명을 지정 가능하고 "조건"에 해당하는 구문과 "동작"에 해당하는 구문을 아이콘, 드랍다운 메뉴와 같은 그래픽 폼을 이용하여 구성할 수 있다. 조건 작성을 위한 필드에는 플랫폼에 현재 등록되어 있는 디바이스 및 외부 서비스를 선택할 수 있으며 해당 디바이스의 속성 값뿐만 아니라 시간 관련 연산을 바탕으로 하는 다양한 조건을 정의할 수 있다. 동작 작성을 위한 필드에서는 플랫폼에 등록된 디바이스/서비스들 중 액츄에이팅이 가능한 속성을 사용자가 변경함으로써 규칙 발동 시 수행해야 할 내용들을 정의할 수 있으며, 단일 디바이스에 대한 속성 변경뿐만 아니라 일괄 점/소등, 웹 인터페이스 내에서의 알람 수신 등의 추가적인 서비스도 동작 내용으로 선언할 수 있다.

```
"if":[{
    "room": "1",
    "device": "weather",
    "property": "weather_today",
    "value":"비",
    "operator":"=="
},{
    //clock
    "type":"onTime",
    "detail":"2018-02-26 07:00",
    "repeat":"None"
}],
"then":[{
    "room" : "3",
    "device" : "12",
    "all_light_off" : "0",
    "bright":"100",
    "color":"rgb(255,255,0)",
    "systemNoti":"0",
    "description":"turn light yellow"
}]
```

그림 5. 웹 인터페이스를 통해 생성한 규칙 데이터
Fig. 5. Rule data generated through web interface

표 5. 규칙 제약의 추가
Table 5. Adding rule constraints

```
// device constraint
rule.constraints.push([
  ifElem.device,
  'm'+ifElem].device,
  'm'+ifElem.device+'.p'+ifElem.property+ifElem.operator+
ifElem.value+''
]);
...
// clock's onTime constraint
rule.constraints.push(
  ['Clock','c','c.timeComp('"+new Date(ifElem.detail)+"')']);
```

그림 5는 웹 인터페이스를 통하여 "2월 26일 07시"에 "날씨가 비"일 경우 "3번 방의 특정 조명 기구를 노란 색으로 점등"하기 위한 규칙 기반 시나리오를 작성했을 때

생성되는 json 데이터를 나타내고 있다. 조건문에는 두 개의 제약 (날씨, 시간)이 정의되어 있으며 별도의 선언이 없으면 AND 조건으로 평가된다. Rule matcher는 fact-base에 새로운 fact가 추가되거나 기존 fact의 내용이 갱신될 때마다 각 rule의 매치 여부를 판단하고 rule의 조건문의 제약들이 모두 만족되면 해당 rule의 "then" 절의 내용을 수행한다. 표 5는 그림 5의 내용으로 표현되는 rule 데이터의 조건문 (if 제약 조건)을 Nools 룰 엔진의 형식에 맞게 변환하여 컴파일하기 위한 과정을 나타내고 있다. 동작문 (then)의 내용 또한 조건문과 유사하게 서버 측에서 수행하고자 하는 내용의 콜백 함수를 동적으로 생성하여 rule-base에 등록하는 형태로 구성된다.

이와 같이, 제안하는 시스템은 웹 인터페이스를 통하여 사용자가 손쉽게 생성한 시나리오의 규칙 관련 데이터를 IoE 룰 엔진에서 요구하는 형식에 맞게 자동으로 변환함으로써 사용자가 어려운 문법을 추가적으로 학습해야 할 필요가 없도록 하였다.

3. 시스템 구현 및 응용 시나리오

3 장에서는 IoE 플랫폼과 Nools 기반 룰 엔진 및 웹 인터페이스의 구현 및 응용 시나리오와 관련된 내용을 보다 자세히 다룬다.

자체 구축되어 있는 IoE 플랫폼은 Node.js 기반의 서버에서 운영되며, 실내조명, 차량 센서 등의 디바이스와 기상 관련 외부 API 서비스가 연동되어 있다. 또한, IoE 룰 엔진 구성을 위해 Nools v0.4.4를 사용하였고, 웹 인터페이스 구축을 위하여 Express 프레임워크와 EJS 템플릿 엔진을 사용하였다. 그림 6은 IoE 플랫폼의 웹 대쉬보드 화면을 나타내고 있다. 플랫폼에서 관리할 수 있는 방의 종류와 수, 외부 서비스 및 각 방에 등록되어 있는 사물의 종류와 개수들을 확인할 수 있다. 예를 들어, 안방의 특정 등을 선택할 경우 그림 7과 같이 현재 해당 사물의 속성 정보를 조회 및 제어할 수 있는 인터페이스를 확인할 수 있다.

그림 6. IoE 플랫폼 웹 대쉬보드
Fig. 6. IoE platform web dashboard

그림 7. IoE 제어를 위한 웹 인터페이스
Fig. 7. Web interface for IoE control

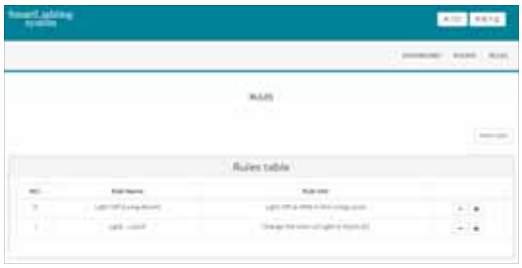

그림 8. IoE 룰 엔진 대쉬보드
Fig. 8. IoE rule engine dashboard

웹 기반의 룰 엔진 인터페이스는 대쉬보드에서 "Rules" 메뉴를 선택하여 확인할 수 있다. 그림 8과 같이, 룰 대쉬보드에서는 현재 플랫폼에 등록되어 있는 룰을 조회할 수 있으며 각 항목은 해당 룰의 이름과 간단한 동작 설명을 표시하고 있다. 룰의 추가는 "Add rules" 버튼을 선택하여 진행 가능하며 그림 9와 같이 룰의 이름과 설명을 입력한 후 조건문과 동작문을 구성한다. 룰의 이름과

설명을 설정하면 조건문을 구성할 수 있는 인터페이스가 그림 10과 같이 나타난다. 그림 10의 예는 앞서 설명한 "매일 아침 7시 날씨가 비일 경우"를 정의하기 위한 모습이다. 그림 10의 상단과 같이 외부 서비스 (weather API)를 선택한 후, 해당 외부 서비스가 제공하는 속성 중 "오늘 날씨"와 비교하고자 하는 값을 정의할 수 있으며, 그림 10의 하단과 같이 시간 관련 설정(2018년 2월 26일 오전 7시 이후 매일 같은 시각) 을 추가할 수 있다.

그림 9. IoE 룰 추가 (이름 및 설명)
Fig. 9. IoE rule add (name and description)

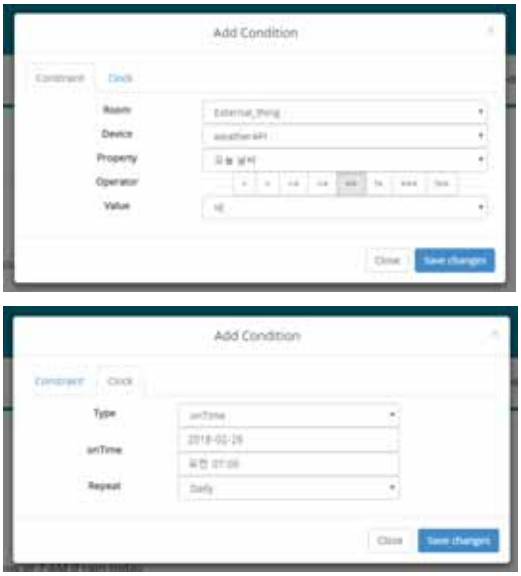

그림 10. IoE 룰 추가 (조건문 구성)
Fig. 10. IoE rule add (defining constraints)

조건문을 추가한 후에는 동작문을 구성할 수 있다. 그림 11은 거실 무드 등을 노란색으로 변경하는 동작을 설정하기 위한 화면으로 드랍다운 메뉴와 속성 슬라이드 바, 색상 선택 다이얼로그 등의 인터페이스 등을 이용하여 보다 직관적으로 제어 대상과 속성을 설정할 수 있다. 조건문과 동작문의 설정이 완료되면 그림 12와 같은 룰 상세 정보 화면이 출력되며 해당 페이지에서 조건문과 동작문을 추가로 구성하거나 기존 항목의 내용을 변경할 수 있다. 모든 설정이 완료된 후 룰을 저장하게 되면 2장 2절에서 설명한 과정을 거쳐 rule의 JSON 데이터들을 Nools 문법에 맞게 변환하여 rule-base에 추가한다. 규칙이 룰 엔진에 추가된 후에는 fact-base에 새로운 fact가 추가되거나 기존 fact의 내용이 갱신되는 경우 rule-base에 존재하는 rule의 매치 여부를 판단하여 상황에 맞는 rule에 정의되어 있는 액션이 수행된다.

그림 11. IoE 룰 추가 (동작문 구성)
Fig. 11. IoE rule add (defining actions)

그림 12. IoE 룰 상세 정보
Fig. 12. IoE rule details

4. 결론 및 향후 연구

본 논문에서는 효율적이고 직관적인 IoE 규칙 기반 시나리오 동작 지원을 위하여 오픈소스 룰 엔진인 Nools 기반의 IoE 룰 엔진과 웹 인터페이스를 제안하였다. 자바스크립트 기반의 범용 패턴 매칭 엔진인 Nools 프레임워크

를 IoE 플랫폼에 등록되어 있는 각종 사물, 사용자, 서비스들 간의 지능형 시나리오에 적합하게 개선하였고 HTML5/CSS3 기반의 웹 인터페이스 구축을 통하여 사용자들이 보다 편리하게 지능형 서비스를 이용할 수 있도록 하였다. 표 6은 제안하는 시스템과 기존 연구들과의 차이점을 시스템 인터페이스 측면, 룰 엔진의 적용 도메인, 룰 생성 방식, 룰 엔진의 커스터마이징 가능 여부 측면으로 정리하여 나타내고 있다. 제안하는 시스템은 HTML5/CSS3의 반응형 인터페이스를 기반으로 모바일 및 웹 브라우저에서 유연하게 활용할 수 있으며, 센서 위주의 룰 생성을 지원하는 기존 연구들과는 달리 센서뿐만 아니라 액추에이터, OpenAPI와 같은 외부 서비스의 속성 및 상태를 룰 생성에 활용할 수 있다. 또한, 룰 생성 과정에서는 디바이스나 서비스의 종류에 따라 룰 생성을 보다 직관적으로 수행할 수 있도록 하는 UI 컴포넌트를 제공하며 Node.JS 기반 오픈소스 룰엔진 프로젝트에 기반하여 IoE 플랫폼에서 추가적으로 필요로 하는 구성요소와 기능들을 손쉽게 추가/확장할 수 있다는 장점들을 보유하고 있다.

향후에는 디바이스 타입에 따른 속성 및 동작의 직관성 향상을 위한 UX 연구를 추가하여 IoE 플랫폼의 사용자 경험을 더욱 향상시키고자 하며, 제안하는 룰엔진과 연계되어 있는 IoE 플랫폼에 축적되는 사용 데이터들을 바탕으로 다양한 룰들을 자동으로 생성하여 사용자에게 추천하기 위한 기술들을 개발하고자 한다.

표 6. 기존 시스템과의 비교
Table 6. Comparison with previous systems

criteria	[5]	[6]	[8]	[11]	Proposed
Interface	web/mobile	web	web	-	web/mobile
Domain	sensor	sensor	sensor actuator	temp. sensor	sensor, actuator, 3rd party service
Rule representation	text	text	-	decision tree	GUI
Customization	△	O	△	△	O

REFERENCES

[1] M. Stauffer, "Connecting the internet of everything," Proceedings of IEEE Hot Chips 26 Symposium, pp. 1-38, 2014.
[2] IBM Watson IoT Platform, http://internetofthings.ibmcloud.com/
[3] Thing+, http://thingplus.net/
[4] AWS IoT, https://aws.amazon.com/iot/
[5] Kyung Hoon Jang and Jae Il Park, "Study on IoT System Design for Factory Site Management Service by Event - Condition - Action Rules", Proceedings of Korean Operations Research And Management Society Conference, pp. 1459~1465, Apr, 2015.
[6] KyeYoung Kim, HyunDong Lee and Dae-Soo Cho, "Real-time Event Processing Role Management System for IFTTT Service," Journal of Korea Multimedia Society, Vol. 20, No. 8, pp. 1379~1386, Aug, 2017.
[7] Jung Min Byun, Sung Soon Park and Kyung Hoon Kim, "Designing and Implementation of Rule-based Automation control and management Framework considering the IoT devices scalability," Proceedings of KIISE Winter Conference, pp. 398~400, Dec, 2015.
[8] Heon-Je Lee, Sung-Soon Park and Kyung-Hoon Kim, "Expandable IoT integrated Management and Rule based automation System," Proceedings of KIISE Winter Conference, pp. 1627~1629, Dec, 2015.
[9] Han-Jo Jeong and Byung-Hwa Park, "Context/Behavior segmentation and Rule-based recommendation system for IoT environment," Proceedings of KIISS Conference, pp. 13~13, Nov, 2015.
[10] Gil Heo, Seoung-Kuk Yang, Mi-Hwa Kim, hyang-jin Lee, Je-Min Kim, Jung-Hwa Choi, Eun-Seok Seo and Young-Tack Park, "A Context-Aware System for Smart Phone Using Rule Base Inference Engine," Proceedings of KIISE Computer Congress, Vol. 27, No. 2C, pp. 273~277, Nov, 2010.
[11] Chang-Seup Han and Chang-Jae Kim, "A Study on Context-Aware IoT Design Using Decision Tree," The Journal of Korean Institute of Information Technology, Vol. 14, No. 1, pp. 203~210, Jan, 2016.
[12] Nools, http://github.com/noolsjs/nools

저자약력

이 성 훈 (Sung-Hun Lee) **[학생회원]**

- 2012년~현재 금오공대 컴퓨터공학과 (학사과정)

<관심분야> 사물인터넷 통신 및 제어, 머신러닝, 빅데이터

황 채 은 (Chae-Eun Hwang) **[학생회원]**

- 2015년~현재 금오공대 컴퓨터공학과(학사과정)

<관심분야> 사물인터넷 통신 및 제어, 웹, 머신러닝등

정 진 우 (Jin-Woo Jeong) **[정회원]**

- 2006년 한양대학교 컴퓨터공학과 (학사)
- 2008년 한양대학교 컴퓨터공학과 (석사)
- 2013년 한양대학교 컴퓨터공학과 (박사)
- 2013년 ~ 2016년 삼성전자 (책임연구원)
- 2016년 ~ 현재 금오공대 컴퓨터공학과 교수

<관심분야> 멀티미디어 정보검색, 데이터마이닝, 사물인터넷 등