반도체디스플레이기술학회지 제17권 제1호(2018년 3월)
Journal of the Semiconductor & Display Technology, Vol. 17, No. 1. March 2018.

SWRL을 이용한 자가 적응 시스템 내에서의 룰 구성

박용범[*†]·안정현[**]
[*†]단국대학교 소프트웨어학과, [**]단국대학교 컴퓨터과 석사과정

Rule Configuration in Self Adaptive System using SWRL

Jung Hyun An[*†] and Young B. Park[**]
[*†]Dankook University Dept. of Software Science, [**]Dankook University Dept. of Computer Science

ABSTRACT

With the development of the Internet of Things technology, a system that ensures the self-adaptability of an environment that includes various IoT devices is attracting public attention. The rules for determining behavior rules in existing self-adaptation systems are based on the assumption of changes in system members and environment. However, in the IoT environment, flexibility is required to determine the behavior rules of various types of IoT devices that change in real time. In this paper, we propose a rule configuration in a self-adaptive system using SWRL based on OWL ontology. The self-adaptive system using the OWL – SWRL rule configuration has two advantages. The first is based on OWL ontology, so we can define the characteristics and behavior of various types of IoT devices as an integrated concept. The second is to define the concept of a rule as a specific language type, and to add, modify and delete a rule at any time as needed. Through the rule configuration in the adaptive system, we have shown that the rule defined in SWRL can provide flexibility and deeper concept expression function to adaptability to IoT environment.

Key Words : Self-adaptive System, SWRL, Ontology, Rule-based Software

1. 서 론

Internet of Thing 기술이 발전함에 따라 다양한 센서와 특징, 목적을 가진 IoT 장비들이 개발되고 있다. 분산되어 있는 IoT 장치 구성원들을 통합하고 분산되어 있는 데이터를 특정한 목적에 사용하기 위한 IoT 플랫폼에서 사용자의 개입 없이 주어진 환경 변화에 대해 시스템 스스로가 환경의 상태를 식별하고 대응 방법을 실행하는 자가 적응 개념이 중요해지고 있다. 유비쿼터스를 실현하기 위한 자가 적응 시스템은 이기종 IoT 장비들로 이루어진 IoT 환경에 대한 자가 적응성을 실현할 수 있어야 한다. 대부분의 자가 적응형 시스템은 자가 적응성을 보장하기 위해 모니터링된 이벤트에 대응하는 방법을 결정하기 위해 암시적 또는 명시적인 룰을 사용한다[1]. 환경 변화에 따라 동작을 바꾸는 장비의 경우 변화를 야기하는 모든 상황을 탐지하고 하부 조형의 룰에 동작을 부가하여 룰이 적용될 때마다 관련된 적응 동작이 활성화 된다[2].

다음의 연구들은 자가 적응형 시스템 내에서의 룰 형식과 모니터링된 환경 변화가 자가 적응 피드백 루프 내에서 룰을 기반으로 하여 처리되고 그로 인해 대응 방법이 피드백으로 전달되는 과정을 자가 적응 시스템 구현을 위한 프레임워크와 스마트 빌딩 시스템에서의 적용을 통해 보여주고 있다[3][4][5].

이기종 장비들을 개념적으로 통합하고 IoT 환경에서 발생하는 데이터들을 자가 적응 시스템과 인간 양 측이 이해할 수 있도록 하기 위해 온톨로지를 룰 구성의 기반

[†]E-mail: ybpark@dankook.ac.kr

이 되는 메타데이터로 활용하는 것이 자가 적응 시스템에서의 룰 구성에 대한 하나의 방안이 될 수 있다. 다음의 연구들은 온톨로지 모델을 룰 구서의 기반으로 사용하여 모니터링 할 환경 변화의 컨텍스트를 인식하고 룰 사용에 온톨로지의 추론 기능을 사용할 수 있다는 것을 시멘틱 웹 환경 상에서의 룰 기반 어플리케이션 구현으로 보여주고 있다[6][7][8].

이러한 온톨로지 기반의 룰 구성 환경에서 OWL-DL 및 OWL-Lite와 Rule Markup 언어(RL)의 조합을 기반으로 한 시멘틱 웹 규칙 언어(SWRL) [9]을 룰을 작성하는데 필요한 주요 언어로 사용할 수 있다. 상호 운용성은 규칙 기반 시스템에 필요한 중요한 기능 중 하나이며 SWRL은 이러한 규칙 언어를 정의하는 중요한 첫 단계이다[10]. 따라서 이 논문에서는 이기종 IoT 환경의 자가 적응 시스템에서 각 장비들 간의 개념적 통합과 시스템의 룰에 유연성과 상호 운용성을 보장하기 위해 SWRL을 기반으로 룰을 작성하고 시스템 내에서 SWRL 형식의 룰을 분석하여 상황에 따라 룰을 생성하거나 삭제, 수정하는 룰 구성을 제안한다.

자가 적응 시스템 내에서 '룰 구성'이라고 정의한 SWRL을 이용한 룰 조작 방법은 기존의 룰 조작에 비해 크게 두 가지 이점을 가진다. 하나는 OWL 온톨로지가 기반이기 때문에 복수의 자가 적응 시스템이 같은 개념의 데이터 모델 개념을 공유할 수 있다는 것이다. 이는 분산된 IoT 환경에서 복수의 자가 적응 시스템으로 상향식 자가 적응 시스템을 구성할 때 보다 유익하다. 다른 하나는 SWRL이라는 정해진 형식으로 룰을 정의하여 시스템이 룰 이외의 상황에 직면하였을 때 혹은 기존의 룰이 환경에 맞지 않다는 판단을 할 때 룰 구성을 쉽게 변경할 수 있다는 것이다.

논문의 나머지 구성은 다음과 같이 되어 있다. 2장에서는 기존의 SWRL로 작성된 룰과 룰을 사용한 시스템에 관련된 기존 연구들을 소개한다. 3장에서는 SWRL 룰의 기반이 되는 OWL 온톨로지 룰셋의 구성과 형식을 설명한다. 4장에서는 SWRL을 이용한 룰의 정의와 자가 적응 시스템의 룰 구성에 관한 내용을 설명하고 프레임워크 내에서의 적용을 보인다. 마지막으로는 결론과 향후 연구를 설명한다.

2. 관련 연구

Zhang와 Hansen의 연구 [11] 는 퍼베이시스 컴퓨팅에서의 자가-관리를 실현하기 위해 동적 컨텍스트 정보가 자가-관리 컨텍스트 온톨로지 집합에서 인코딩 되는 시멘틱 웹 기반 자가 관리 접근법을 제안하고 있다. 퍼베이시스 온톨로지 컨텍스트 집합에서 컨텍스트 흐름을 제시하고 SWRL을 이용한 자가-관리 룰을 미들웨어에 적용한다.

FRAMESELF [12] 는 널리 분산되어 있으며 환경 변화에 따라 자주 발전하는 수천종의 이기종 시스템을 연결하는 Merchine-to-Merchine 시스템 내에서 M2M 통신의 복잡성을 해소하기 위해 설계된 온톨로지 기반 프레임워크이다. 서비스 지향 및 이벤트 위주의 통신 패턴을 제어하기 위해 SWRL을 이용하여 이벤트 룰을 정의하고 확장 가능한 제어 루프 구성 요소 다이어그램을 제시하고 있다.

다음의 연구들은 SWRL의 표준화를 통해 프레임워크 내에서의 추론기 사용의 범용성 확보와 SWRL로 이루어진 온톨로지 룰셋의 구성, 룰셋 내에서의 룰 관리, 시스템 내에서의 룰 적용에 대한 요구사항과 아키텍처를 정의하고 이를 이용한 실제 소프트웨어 내에서의 룰 구성에 대한 적용 결과를 설명하고 있다[13][14][15]. Martin 과 Holger의 연구[13]은 시맨틱 웹 환경에서 규칙 기반 시스템 간의 상호 운용성을 확보하기 위해 OWL 개념에 따라 표현된 Hornkile 규칙을 작성하여 OWL Individual을 추론할 수 있는 SWRL 룰 형식을 지정하고, SWRL 편집기를 설계하여 Protégé OWL 플러그인 내에서 작동하는 Jess 규칙 엔진과 편집기를 통한 온톨로지와 RuleML의 통합 언어가 상호 운용성을 보장할 수 있음을 보여준다.

Domenico와 Luigi의 연구[14]는 주석에 의해 자동 검색, 구성 및 호출을 가능하기 하는 시멘틱 웹 서비스 내에서 OWL-S 원자 프로세스에 SWRL을 도입하였다. 도메인 룰을 SWRL로 지정함으로 인해 IOPR(Inputs, Outputs, Preconditions and Result) 모델을 표현하여 SWRL 룰을 통한 모델 표현이 가능함을 보여주었다. SWRLAPI[15]는 Protégé-OWL 온톨로지 개발 툴킷의 확장 프로그램으로 룰 개발을 위한 저작 환경과 규칙 기반 응용 프로그램 구축을 지원하는 일련의 응용 프로그래머 인터페이스를 제공한다. API는 OWL 온톨로지 쿼리에 사용되는 SWRL과 다양한 정보 형식 간의 상호 운용 도구를 지원한다. SWRLAPI를 통해 실제 소프트웨어 내에서 SWRL을 이용한 룰 작성과 작성된 룰이 실제로 적용될 수 있음을 보여준다.

OBALS [16] 는 적응형 하이퍼 미디어에 의해 친숙한 사용자 인터페이스를 지원하는 적응형 이-러닝 시스템을 위한 프레임워크이다. 학습 온톨로지 모델을 형성하고 적응 학습 전략에 관한 룰을 SWRL로 정의하여 생성된 프레임워크 내에서의 자가 적응성을 보여준다. Vassileva와 Bontchev의 연구[17]는 자가 적응 프로세스에 대해 유연하고 관리 가능한 제어를 용이하게 하는 적응 엔진을 개발하기 위한 개념과 요구사항을 제안하고 있다. SWRI을 이용한 자가 적응 룰과 적응 프로세스를 제시하고 규칙 기반 접근법을 통한 플랫폼 구축으로 SWRI의 실제 구현과

그 결과를 보여준고 있다.

온톨로지 모델을 이용하면 분산 환경에서의 개념 표현에 통합된 기준을 제공하면서도 유연성과 신뢰성을 부여할 수 있다. Ming과 Jinglon의 연구[18]는 분산 환경에서의 멀티-레이어 프레임워크에 온톨로지 모델을 도입하여 IoT-스마트 홈 어플리케이션을 대상으로 전체 시스템의 신뢰성을 부여할 수 있음을 보여주었다. 또한 Diego의 연구[19]는 IoT 장비의 특성을 고려한 배치 문제를 해결하기 위해 전체 환경의 목표와 각 장비의 특징을 온톨로지 모델로 생성하였다. 그리고 이를 통해 상호 운용 가능한 시스템 내에서의 프레임워크를 도입하여 온톨로지의 유용성을 입증하였다.

3. OWL 룰셋과 룰 구성 프로세스

이 장에서는 논문에서 제안하는 룰 구성을 적용하기 위해 IoT 환경을 대상으로 한 자가 적응 시스템의 구조와 룰 구성이 작동하는 다중 피드백 자가 적응 시스템의 흐름, 그리고 룰 구성에서 사용되는 온톨로지 룰셋과 SWRL 룰 형식에 대해 설명한다. 룰 구성의 적용 대상이 되는 IoT 환경에 대한 자가 적응 시스템은 다중 피드백 루프 구조의 자가 적응 시스템이다. 해당 자가 적응 시스템은 복수의 이기종 IoT 장비로 이루어진 하나의 IoT 환경에 대한 자가 적응성을 담당한다. 또한 상위 피드백 루프는 IoT 환경들로 이루어진 IT 에코시스템의 자가 적응성을 담당한다. 각각의 하위 자가 적응 시스템들은 온톨로지로 구성된 룰셋을 가지고 있으며 룰셋 안의 SWRL을 참조하여 피드백 루프의 각 요소들에 대해 행동 기준을 결정한다.

초기 단계에서 상위 자가 적응 시스템은 온톨로지 룰셋을 파일 형태로 하위 자가적응 시스템에 배포한다. 상위 자가적응 시스템이 배포하는 룰셋은 해당 자가적응 시스템에 대한 기초적인 공리와 룰들을 포함하고 있다. 각 하위 자가 적응 시스템들은 평소에는 상위 자가 적응 시스템이 배포한 초기 룰셋에 따라 발생한 문제에 대한 자가 적응 솔루션을 실행한다. 자가적응 솔루션을 실행하던 중 자가 적응 시스템이 가지고 있는 룰 중에서 솔루션을 발견하지 못할 경우 하위 자가적응 시스템은 상위 자가적응 시스템에 해당 상황을 전달한다.

상위 자가적응 시스템은 요청 받은 상황에 맞는 룰을 생성하여 해당 하위 자가적응 시스템에 전달하게 된다. 하위 자가적응 시스템은 전달받은 룰을 분류하여 객체에 저장한 후 룰셋에 반영한다. Fig. 1은 상위 자가적응 시스템으로부터 하위 자가적응 시스템으로의 룰 구성에 대한 전체 흐름을 보여준다.

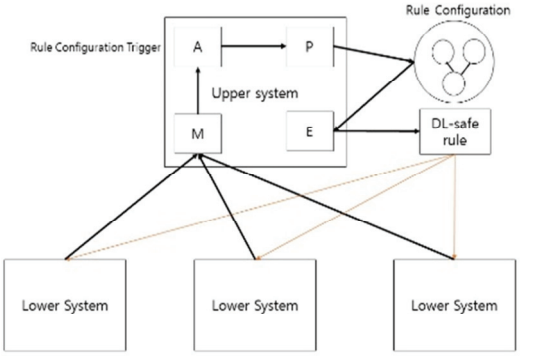

Fig. 1. Rule configuration process in self-adaptive system about IoT environment.

IoT 환경을 대상으로 한 자가 적응 시스템 내에서 사용되는 온톨로지 모델은 OWL-DL 수준의 표현력을 가진 온톨로지이다. OWL-DL 온톨로지는 IoT 환경 내의 이기종 장비들의 개념을 통합하고 개념 관계에 대한 추론 기능을 지원하면서도 SWRL을 적용할 수 있을 만한 체계적인 표현 제한을 제공하기 때문에 자가 적응 시스템 내에서의 룰셋 모델로 선택하였다. 일반적으로 룰셋은 룰 기반 시스템 내에서의 행동 방침을 결정하는 룰들의 집합을 의미한다. 하지만 이 논문에서는 온톨로지를 기반으로 한 SWRL로 작성된 룰들의 집합과 SWRL 추론기 내의 추론 기능을 보조하기 위한 온톨로지의 공리들을 포함한 SWRL 룰의 집합을 룰셋이라고 지칭한다.

IoT 환경에 대한 자가 적응 시스템에서 사용하는 룰셋 온톨로지는 functional syntax의 형식으로 작성되었다. OWL2 XML syntax나 맨체스터 syntax가 아닌 functional syntax로 작성된 온톨로지 모델을 사용하는 이유는 functional syntax가 자가 적응 시스템 내에서 SWRL 룰의 구조를 정의하고 룰 구성이 발생하였을 때 룰과 온톨로지 공리를 추가하는데 유리한 직관적인 표현력을 제공하기 때문이다. Functional syntax로 작성된 온톨로지 모델은 크게 세가지 부분으로 구성되어 있다. Prefix 부분은 온톨로지의 종류와 제약 명세가 기록되어 있는 온톨로지 모델의 특징을 나타낸다. 온톨로지 공리 부분은 SWRL 룰이 자가 적응 시스템 내에서 추론기를 사용하여 조건을 판단하기 위한 근거를 제공한다. SWRL 룰은 룰셋 내에서 룰 자체를 표현한 부분으로 자가 적응 시스템 내에서 모니터링 정보를 통해 문제를 식별하고 자가 적응 대상 환경의 상태를 식별하는 데 사용된다. Fig. 2는 IoT 환경에 대한 자가 적응 시스템에서 사용되는 전체 OWL-DL 온톨로지 모델의 functional syntax 표현을 보여준다.

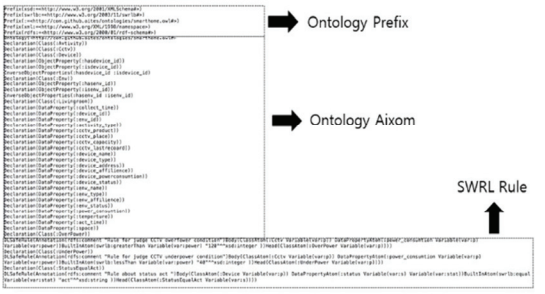

Fig. 2. OWL ontology model in self-adaptive system.

자가 적응 시스템 내에서의 온톨로지 모델은 대상 환경과 구성원들의 특징을 대표한다. 이 온톨로지 모델은 얼마든지 XML syntax와 맨체스터 syntax로 변환할 수 있으며 일반적인 온톨로지 프레임워크에서 사용할 수 있다. IoT 환경에 대한 자가적응 시스템이 사용하는 온톨로지 모델은 자가 적응 시스템의 환경과 참여 IoT 장비들, 그리고 환경과 구성원의 상호 작용으로 발생하는 데이터, 자가 적응 시스템 내의 각 피드백 루프에서 발생하는 데이터의 개념을 담을 수 있는 공리와 룰들로 구성되어 있다. Fig. 3은 자가-적응 시스템에서 사용되는 전체 온톨로지 모델에 대한 그래프 표현을 보여준다.

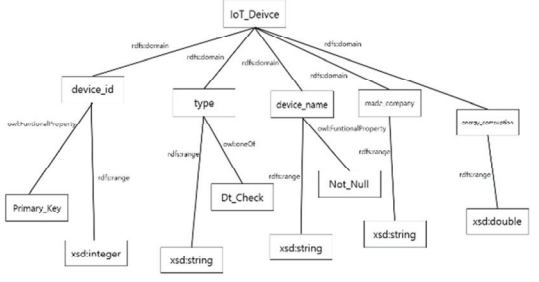

Fig. 3. A graph of ontology model applied to self -adaptation system.

온톨로지 모델 룰셋에 있는 SWRL 룰은 DL-안전 룰의 여건을 만족하고 있다. DL-안전 룰이란 룰의 공리가 온톨로지의 individual에만 적용되는 룰 제한 조건이다. 이는 온톨로지 내에서 표현하지 못하는 추론을 SWRl이 표현하지 못하도록 제한하는데 중요한 역할을 한다. 온톨로지 룰셋에 적용된 룰은 특정한 형식으로 분류할 수 있다. Annotation 절은 Rule의 내용을 사용자가 이해할 수 있는 형식으로 표현해 주는 역할을 한다. Body 절 밑에는 DataPropertyAtom과 BuiltInAtom이 존재한다. 전자는 룰 내에서 어떤 DataPropery 공리를 사용할 것인지를 명시한다. 후자는 실제 rule이 적용될 공리를 나타낸다. 마지막으로 Head 절의 ClassAtom은 이 룰에 적용될 individual이 어떤 클래스에 소속되는가를 명시한다. 하위 자가 적응 시스템 내에서 부분별로 파싱된 룰은 상위 자가 적응 시스템의 요청에 의해 일부를 변경시켜 적용할 수 있다.

4. 자가 적응 프레임워크 내에서의 룰 구성 구현

이 장에서는 자가적응 프레임워크 내에서 미리 선언된 룰 객체 내에 룰을 불러오고 상위 시스템의 시그널에 의한 하위 자가 적응 시스템의 룰 구성이 이루어지는 과정을 설명한다. 앞선 장에서 설명했듯이, functional syntax로 작성된 DL-안전 룰의 구성 부분들은 일정한 형식에 따라서 나눌 수 있다. IoT 환경에 대한 자가 적응 프레임워크는 이 룰의 구성을 분류하고 룰을 객체 내로 불러와 자가적응 프레임워크 내에서 사용할 수 있게 해주는 일련의 기능들을 제공한다. 자가 적응 프레임워크 내에서의 룰 구성은 크게 룰을 OWL 프레임워크 내에 로드하고 SWRL 추론기를 사용해 컨텍스트에 룰이 적용되는지를 판단하는 부분과 자체적으로 룰을 파싱하고 파싱된 룰에 각 부분을 조절하기 위한 부분으로 나누어진다. 상위 자가 적

```
private void headParser(String atomHead){
    String[] parser = atomHead.split("\\s");
    String ruleName = parser[0].substring(16);
    this.ruleName = ruleName;
}
private void classAtomParser(String classAtomBody){
    String[] parser = classAtomBody.split("\\s");
    String classAtom = parser[0].substring(11);

    String variable = parser[1].substring(13, parser[1].length()-2);
    rule = new Rule(ruleName,classAtom,variable);
}
private void propertyAtomParser(String propertyAtom){
    String[] parser = propertyAtom.split("\\s");
    String[] variables = new String[2];
    int typeIndex = propertyAtom.indexOf("(");
    String type = parser[0].substring(0, typeIndex);

    String property = parser[0].substring(typeIndex+2);

    for(int i=1; i<parser.length; i++){
        if(i == parser.length-1){
            String variable = parser[i].substring(13, parser[i].length()-2);

            variables[i-1] = variable;
        }
        else{
            String variable = parser[i].substring(13, parser[i].length()-1);

            variables[i-1] = variable;
        }

    }
    rule.addPropertyAtom(type, property, variables);
}
private void builtInAtomParser(String builtInAtom){
    String[] parser = builtInAtom.split("\\s");
    String[] variables = new String[1];
    int typeIndex = builtInAtom.indexOf("(");
    String type = parser[0].substring(0, typeIndex);

    String property = parser[0].substring(typeIndex+7);


    String variable = parser[1].substring(13, parser[1].length()-1);

    variables[0] = variable;
    String atom = parser[2].substring(0, parser[2].length()-1);
    rule.addBuiltInAtom(property, variables, atom);
```

Fig. 5. Rule parsing implementation code in the self-adaptive system framework.

응형 시스템에서는 주로 초기 룰셋 생성과 룰 조정, 생성된 룰셋과 룰을 하위 자가 적응 시스템으로 전달하는 기능을 사용한다. 하위 자가적응 시스템에서는 룰셋을 로드하고 그 안의 룰을 추론기를 이용하여 사용하는 기능을 사용한다. Fig. 4는 실제 자가적응 프레임워크에서 구현된 룰 파싱에 관련된 코드를 보여준다.

IoT 환경에 관한 자가 적응 프레임워크가 자바 기반으로 작성되었기 때문에 룰 구성에 관련된 기능들도 자바를 기반으로 구현되어 있다. 룰에 관련된 주요 기능은 문자열을 기반으로 하여 문자열의 구분 문자를 기반으로 룰을 분류하고 룰을 문자열 내에서 사용하기 때문에 범용성과 함께 복잡한 처리 절차 없이 룰 구성을 사용할 수 있도록 해준다.

5. 결론 및 향후 연구

이 논문에서는 IoT 환경에 대한 자가 적응 시스템 내에서 functional syntax로 작성된 OWL 온톨로지 ruleset의 구성과 SWRL로 작성된 rule을 시스템이 파싱하여 상황 변화에 따라 새로운 룰을 추가하거나 삭제, 수정하는 Rule configuration을 설명하였다. 또한 실제 자가 적응 시스템 내에서의 Ruleset과 Rule의 구성을 보이고 프레임 워크 내에서 Rule configuration이 어떻게 동작하는지를 설명하였다.

Rule configuration은 자가 적응 시스템 내에서 룰이 사용될 때 초기 룰에 없거나 부적절한 상황이 발생하였을 시 룰 자체를 수정하기 위해 발생한다. Rule configuration은 크게 룰 변경 요청 확인과 요청에 의한 룰 변경, 변경된 룰을 룰 셋에 적용하는 단계로 나누어지며 자가 적응 시스템에 상호 운용성과 상황에 따른 더 깊은 수준의 자가 적응성을 부여해 줄 수 있다.

논문에서는 SWRL의 기초적인 공리를 이용한 룰의 생성과 삭제, 수정에 관련된 내용에 초점을 맞추었다. 이 내용을 기반으로 복잡한 조건에서의 룰 구성은 기초적인 룰을 중첩하여 적용하는 것으로 구현할 수 있다. 앞으로의 연구에서는 복합적인 룰 생성과 삭제 시에 룰 간의 관계 공리를 처리하는 방법에 대하여 연구한다.

감사의 글

"본 연구는 미래창조과학부 및 정보통신기술진흥센터의 고용계약형 SW석사과정 지원사업의 연구결과로 수행되었음"(H0116-16-1015).

"이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 지역신산업 선도인력양성사업성과임"(No.NRF-2016H1D5A1909989).

참고문헌

1. Wang, Qianxiang. "Towards a rule model for self-adaptive software." *ACM SIGSOFT Software Engineering Notes* 30.1 (2005): 8.
2. Neto, João José. "Adaptive rule-driven devices-general formulation and case study." *International Conference on Implementation and Application of Automata*. Springer, Berlin, Heidelberg, 2001.
3. Dashofy, Eric M., André Van der Hoek, and Richard N. Taylor. "Towards architecture-based self-healing systems." *Proceedings of the first workshop on Self-healing systems*. ACM, 2002.
4. Guillemin, Antoine, and Nicolas Morel. "An innovative lighting controller integrated in a self-adaptive building control system." *Energy and buildings,* 33.5 (2001): 477-487.
5. David, Pierre-Charles, and Thomas Ledoux. "Towards a framework for self-adaptive component-based applications." *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, Berlin, Heidelberg, 2003.
6. Wache, Holger, et al. "Ontology-based integration of information-a survey of existing approaches." *IJCAI-01 workshop: ontologies and information sharing*. Vol. 2001. 2001.
7. Golbreich, Christine. "Combining rule and ontology reasoners for the semantic web." *International Workshop on Rules and Rule Markup Languages for the Semantic Web*. Springer, Berlin, Heidelberg, 2004.
8. Christopoulou, Eleni, Christos Goumopoulos, and Achilles Kameas. "An ontology-based context management and reasoning process for UbiComp applications." *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*. ACM, 2005.
9. Horrocks, Ian, et al. "SWRL: A semantic web rule language combining OWL and RuleML." *W3C Member submission* 21 (2004): 79.
10. O'connor, Martin, et al. "Supporting rule system interoperability on the semantic web with SWRL." *International Semantic Web Conference*. Springer, Berlin, Heidelberg, 2005.
11. Zhang, Weishan, and Klaus Marius Hansen. "Semantic web based self-management for a pervasive service middleware." *Self-Adaptive and Self-Organizing Systems, 2008. SASO'08. Second IEEE International Conference on*. IEEE, 2008.
12. Alaya, Mahdi Ben, and Thierry Monteil. "FRAMESELF: an ontology-based framework for the self-management of machine-to-machine systems." *Concurrency and*

Computation: Practice and Experience 27.6 (2015): 1412-1426.
13. O'connor, Martin, et al. "Writing rules for the semantic web using SWRL and Jess." *Protégé With Rules WS, Madrid* (2005).
14. Redavid, Domenico, et al. "OWL-S atomic services composition with SWRL rules." *International Symposium on Methodologies for Intelligent Systems*. Springer, Berlin, Heidelberg, 2008.
15. O'Connor, Martin J., et al. "The SWRLAPI: A Development Environment for Working with SWRL Rules." *OWLED*. 2008.
16. Min, Wang Xiao, Cui Wei, and Che Lei. "Research of ontology-based adaptive learning system." *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on*. Vol. 2. IEEE, 2008.
17. Vassileva, Dessislava, and Boyan Bontchev. "Adaptation Engine Construction based on Formal Rules." *CSEDU (1)*. 2009.
18. Tao, Ming, et al. "Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes." *Future Generation Computer Systems* 78 (2018): 1040-1051.
19. Sánchez-de-Rivera, Diego, et al. "Adaptation of ontology sets for water related scenarios management with IoT systems for a more productive and sustainable agriculture systems." (2017).

접수일: 2018년 2월 6일, 심사일: 2018년 3월 20일,
게재확정일: 2018년 3월 21일