JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 26 No. 12, pp. 151-158, December 2021
https://doi.org/10.9708/jksci.2021.26.12.151

Development of Metrics to Measure Reusability of Services of IoT Software

Eun-Sook Cho*

*Professor, Dept. of Software Engineering, Seoil University, Seoul, Korea

[Abstract]

Internet of Things (IoT) technology, which provides services by connecting various objects in the real world and objects in the virtual world based on the Internet, is emerging as a technology that enables a hyper-connected society in the era of the 4th industrial revolution. Since IoT technology is a convergence technology that encompasses devices, networks, platforms, and services, various studies are being conducted. Among these studies, studies on measures that can measure service quality provided by IoT software are still insufficient. IoT software has hardware parts of the Internet of Things, technologies based on them, features of embedded software, and network features. These features are used as elements defining IoT software quality measurement metrics. However, these features are considered in the metrics related to IoT software quality measurement so far. Therefore, this paper presents a metric for reusability measurement among various quality factors of IoT software in consideration of these factors. In particular, since IoT software is used through IoT devices, services in IoT software must be designed to be changed, replaced, or expanded, and metrics that can measure this are very necessary. In this paper, we propose three metrics: changeability, replaceability, and scalability that can measure and evaluate the reusability of IoT software services were presented, and the metrics presented through case studies were verified. It is expected that the service quality verification of IoT software will be carried out through the metrics presented in this paper, thereby contributing to the improvement of users' service satisfaction.

▸ **Key words**: IoT Software, Service Quality, Reusability, Changeability, Replaceability, Extensibility

• First Author: Eun-Sook Cho, Corresponding Author: Eun-Sook Cho
*Eun-Sook Cho (escho@seoil.ac.kr), Dept. of Software Engineering, Seoil University
• Received: 2021. 11. 08, Revised: 2021. 12. 06, Accepted: 2021. 12. 06.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

[요 약]

인터넷을 기반으로 실세계에 존재하는 여러 사물들과 가상 세계에 있는 사물들이 연결되어 서비스를 제공하는 사물인터넷(IoT) 기술이 4차 산업혁명 시대의 초연결 사회를 가능하게 하는 기술로 부각되고 있다. 사물 인터넷 기술은 디바이스, 네트워크, 플랫폼, 서비스를 아우르는 융합 기술이기 때문에 여러 다양한 연구들이 진행되고 있다. 이러한 연구들 중에 IoT 소프트웨어가 제공하는 서비스 품질을 측정할 수 있는 척도들에 관한 연구는 아직 많이 미흡한 실정이다. IoT 소프트웨어는 사물인터넷이 가지는 하드웨어 부분과 이를 바탕으로 하는 기술, 임베디드 소프트웨어가 가지는 특징, 네트워크의 특징 들을 가지고 있다. 이러한 특징들은 IoT 소프트웨어 품질 측정 메트릭을 정의하는 요소로 활용된다. 그러나 현재까지의 IoT 소프트웨어 품질 측정 관련 메트릭들에서는 이러한 특징들을 고려하고 있지 있다. 따라서 본 논문에서는 이러한 요소들을 고려하여 IoT 소프트웨어의 여러 가지 품질 요소 가운데 재사용성 측정을 위한 메트릭을 제시한다. 특히 IoT 소프트웨어는 사물인터넷 디바이스를 통해 활용되기 때문에 IoT 소프트웨어 내 서비스가 변경이나 교체 또는 확장이 가능하도록 설계되어야 하며, 이를 측정할 수 있는 메트릭이 매우 필요하다. 따라서, 본 논문에서는 IoT 소프트웨어의 서비스들에 대한 재사용성을 측정 및 평가할 수 있는 변경성, 교체성, 확장성이라는 3가지 메트릭을 제시하고, 사례연구를 통해 제시한 메트릭에 대한 검증을 하였다. 본 논문에서 제시한 메트릭을 통해 IoT 소프트웨어의 서비스 품질 검증이 이루어짐으로써 사용자들의 서비스 만족도 향상에 기여할 수 있을 것이라 기대한다.

▸ **주제어**: 사물인터넷 소프트웨어, 소프트웨어 품질, 재사용성, 변경가능성, 교체성, 확장성

I. Introduction

인터넷 기술이 발전함에 따라 시간, 장소 등에 구애받지 않고 정보를 검색할 수 있을 뿐만 아니라 이를 활용하여 새로운 정보나 서비스를 제공하거나 응용할 수 있는 여러 다양한 방법이나 기술들이 등장하고 있다. 이러한 유형 중 하나가 사물 인터넷 기술(IoT: Internet of Thing)로서, 이는 실세계에 존재하는 사물의 영역까지 인터넷 기술을 적용한 기술로서, 인터넷 기반에서 사물과 사람, 사물과 사물 간에 서로 정보 교환을 통해 데이터를 수집 및 저장하는 지능형 기술 서비스라고 할 수 있다[1].

IoT는 무선 센서 네트워크(WSN: Wireless Sensor Network)이나 M2M(Machine-to-Machine) 등과 같은 기술들을 반영하면서 인터넷 기술을 적용하여 다양한 서비스들을 창출하고 있다. 따라서 최근에는 IT 분야 뿐만 아니라 가전, 자동차, 엘리베이터, 가구, 의료기기 등 여러 다양한 분야에 IoT 기술이 적용되어 서비스들을 제공하고 있다[2].

미국 국가정보위원회 (NIC, National Intelligence Council)에서는 정치, 경제, 교육, 군사 등 여러 분야에서 국가 경쟁력에 영향을 끼칠 수 있는 기술로 IoT 기술을 선정하였듯이, 이처럼 IoT는 경제 및 산업 전반에 걸쳐서 막대한 파급 효과를 미칠 것으로 기대하고 있다[3].

또한 IoT 소프트웨어의 특성은 다른 일반적인 소프트웨어와 달리 사람의 개입 없이 기기 홀로 다른 기기들과 상호 연동한다는 것이다. 그러나 현재까지의 소프트웨어 품질 측정 관련 메트릭에서는 이러한 특징을 고려하지 않고 있다. 따라서 본 논문에서는 이러한 요소들을 고려하여 IoT 기반 소프트웨어의 특성을 반영한 재사용성 측정 메트릭을 제시한다. 특히 IoT 소프트웨어의 여러 가지 품질 요소 가운데 재사용성 측정을 위한 메트릭들을 제시한다.

본 논문의 구성은 2장에서는 관련연구로 IoT 소프트웨어 관련 기존 품질평가 연구 방법들에 대해 설명하며, 3장에서는 IoT Software의 특징들을 제시하고, IoT 소프트웨어의 서비스 재사용성 측정을 위한 메트릭들을 제안한다. 4장에서는 3장에서 제안한 기법을 사례 연구한 결과를 제시한다. 마지막으로 5장에서는 결론과 향후 연구 과제를 제시한다.

II. Preliminaries

1. Related works

1.1 Measurement Method of IoT's Quality

[4]의 연구는 IoT 환경의 서비스 향상을 위한 품질 측정 방법으로 품질관리, 운용관리, 그리고 프로토콜 취약성에 대처하는 측정모델을 제시하였다. 여기서는 품질평가를 하기 위해 단일 통합 테이블에 기능과 품질 요구사항을 취합하였다. 그러나 이 연구는 IoT의 구조 및 기능에 초점을 두었기 때문에 측정항목이 부족하고, 품질항목과 세부항목에 대한 설계나 내용이 부족하다.

[5,6]의 연구에서는 IoT 특징들을 도출한 후 이를 기반으로 한 품질 평가 모델을 제시하고, 공존성, 상호 운영성, 복구성, 조작성, 사용가능성 등에 대한 메트릭을 정의했다. 그러나 이 연구는 사물인터넷의 하드웨어적인 작동에 초점을 둔 메트릭을 제시했다. 따라서 IoT 소프트웨어 측면의 품질을 측정하기 위한 메트릭에 대한 내용은 미흡하다.

1.2 Reusability Metric by Heuristic Method

경험적 방법에 의한 재사용성 측정 방법으로는 Halstead의 프로그램의 크기를 이용한 방법, McCabe의 Cyclomatic 복잡도를 이용한 방법, 정규도(Regularity)와 재사용 빈도수에 의한 소프트웨어의 재사용성을 측정하는 방법들이 있다[7]. 그러나 이러한 방법은 일반적인 소프트웨어 재사용성에는 쉽게 적용되지만, 컴포넌트, 프레임워크, 그리고 IoT 소프트웨어의 서비스에 대한 재사용성을 측정하는데 있어서는 IoT 서비스가 갖는 하드웨어 디바이스와의 공존성, 이동성, 실시간 연동성, 상호작용 등과 같은 특징들이 반영되어 있지 않다.

1.3 Reusability Metric by Qualitative Method

정성적 측정 방법으로 ISO/IEC 9126은 소프트웨어의 품질을 6개의 특징으로 나누었고, 6개의 특징들 별로 하위 특징들이 존재한다[8]. 이러한 하위 특징들은 내부 측정기준과 외부 측정기준으로 측정한다. 그러나 ISO/IEC 9126에서 제시하는 측정 모델은 일반적인 소프트웨어 품질 측정에 관한 모델은 제시하고 있지만 IoT 소프트웨어와 관련된 특징들에 대해서는 미반영된 부분들이 존재하기 때문에 그대로 적용하기가 어렵다.

In Guen[9]의 연구에서는 재사용성 측정방법으로 2가지 방법인 직접 측정과 간접 측정 방법을 제시하였다. 직접 측정은 컴포넌트를 구성하는 클래스들과 컴포넌트의 인터페이스들을 기반으로, 컴포넌트 크기, 복잡도, 결합도, 응집도 등을 재사용성 측정 요소로 반영하였다. 간접 측정은 컴포넌트의 이해도, 적용성, 변경성, 모듈화 등을 요소로 재사용성을 측정 하는데 사용하였다. 그러나 이 방법에서도 IoT 소프트웨어의 특징을 고려한 확장성이나 이식성에 대한 측정 방법은 제시하지 못했다.

재사용성 측정 방법과 관련된 기존의 연구들의 한계점들은 [Table 1]과 같다.

Table 1. Limitations of Existing Researches

Research Types	Limitations
[4]'s Research	• focus on IoT's structure and functions • lack measurement items
[5,6]'s Research	• focus on IoT's hardware aspects
[7]'s Research	• measure general software's reusability • don't consider characteristics of IoT service
[8]'s Research	• don't reflect IoT software's reusability items
[9]'s Research	• don't propose extensibility or portability of IoT software

III. The Proposed Scheme

이 장에서는 IoT 소프트웨어의 재사용성을 향상하기 위해 IoT 소프트웨어의 특징들을 제시하고, 이를 토대로 재사용성 관련 품질 속성들을 도출하고, 각 특성 별 메트릭들을 제시한다.

1. Characteristics of IoT Software

IoT 소프트웨어는 사물인터넷이 가지는 하드웨어 부분과 이를 바탕으로 하는 기술, 임베디드 소프트웨어가 가지는 특징, 네트워크의 특징 들을 가지고 있다. 이러한 특징들은 IoT 소프트웨어 품질 측정 메트릭을 정의하는 요소로 활용된다. [10]의 연구에 따르면 이러한 특징들을 다음과 같이 7가지로 도출하여 제시하고 있다.

1.1. Mobility

이동성은 사용자가 사물인터넷 기기를 이용해서 환경이나 공간에 제약 없이 사물인터넷에서 제공하는 서비스를 지속적으로 제공 받을 수 있는가를 뜻한다. 사물인터넷 기기를 통해 사물인터넷에 저장된 정보를 네트워크 상에서 자유롭게 주고받을 수 있다[11].

1.2. Coexistence with Hardware

하드웨어와의 공존성은 사물인터넷이 기존 임베디드 소프트웨어의 특성을 기반으로 하기 때문이다. 사물 인터넷이 컴퓨팅 기능을 담당하는 스마트 한 '사물'이 되기 위해

서는 운영체제 및 네트워크를 담을 수 있는 하드웨어가 필요하다. 주요 하드웨어로는 센서 장치, 처리 장치, 송수신 장치, 전원 장치 등이 포함된다[11].

1.3. Real-time Accessibility

실시간 접근성은 장소와 시간에 제약받지 않고 언제나 서비스에 접근 가능한 특징을 말한다. 네트워크 환경 하에서는 실시간으로 사물과 사물 간에 통신이 되어야 하고, 사용자는 실시간으로 데이터나 서비스에 대한 접근이 가능해야 한다.

1.4. Maintenance and Security of Data

데이터의 유지 및 보안은 사물인터넷이 데이터를 수집하고 처리하는 과정에서 데이터의 무결성이 유지되고 외부 공격으로부터 보호되어야 하는 특징을 말한다. IoT 는 단말장치, 센서 네트워크, 서버 등에서 보안 위협이 발생할 수 있으므로 데이터의 유지 및 보안에 대한 대응책이 마련되어 있어야 한다[11].

1.5. Providing Screens

화면 제공성은 사물인터넷이 수집한 데이터를 사용자에게 화면을 통해 제공한다는 의미이다. 사용자는 스마트폰, 스마트 워치 등의 다양한 디바이스 화면을 통해 언제 어디서나 필요로 하는 정보를 검색 및 수정 할 수 있어야 한다[12].

1.6. Connection Reliability

연결 신뢰성은 네트워크 환경 하에서 IoT를 통해 수집된 데이터를 데이터베이스에 저장시키는 미들웨어 시스템의 연결 상태를 말한다. 블루투스, NFC, 와이파이 등과 같은 다양한 형태의 네트워크 연결로 구성되는 사물인터넷은 센서 게이트웨이를 통해 지속적인 운영을 보장해야 한다.

1.7. Interoperability

상호연동성은 사물인터넷이 다른 사물인터넷과 플랫폼에 독립적으로 서로 연동되어 정보를 주고받을 수 있는 특징을 말한다. 예를 들어, 여러 디바이스들이 스마트폰이나 태블릿 PC 등과 연결되어 정보를 주고 받을 수 있어야 한다.

2. Metrics for Measuring Reusability of IoT Software

ISO/IEC 25000을 기반으로 IoT 소프트웨어의 품질 속성들을 분류할 수 있으나, 해당 기준은 메트릭 정의가 명확하지 않다[12]. 본 논문에서는 보편화 된 ISO/IEC 9126을 토대로 IoT 소프트웨어가 지니는 품질 특성에 따라 메트릭을 새롭게 정의했다. 특히 본 논문에서 제시하는 메트릭은 재사용성에 초점을 맞추어 정의하였다.

2.1. A Metric for Measuring Changeability of IoT Software

IoT 소프트웨어에서의 변경성(Changeability)이라 함은 IoT 소프트웨어 내의 서비스들 가운데 변경 가능한 서비스들의 비율을 의미한다[13,14].

[정의 1] 변경성(RSC: Rate of Service Changeability)

$$RSC(n) = \begin{cases} 1 - \dfrac{S_c(n)}{S(n)} & (S(n) > 0) \\ 0 & (otherwise) \end{cases}$$

where,

Sc(n): IoT 소프트웨어의 변경 가능한 서비스 갯수

S(n): IoT 소프트웨어의 서비스 총 개수

2.2 A Metric for Measuring Replaceability of IoT Software

IoT 소프트웨어에서의 교체성(Replaceability)이라 함은 IoT 소프트웨어 내의 서비스들 가운데 변경 가능한 서비스들의 비율을 의미한다[13,14].

[정의 2] 교체성(RSR: Rate of Service Replaceability)

RSR(n)는 IoT 소프트웨어 내의 서비스들 가운데 교체 가능한 서비스들의 비율을 의미한다.

$$RSR(n) = \begin{cases} 1 - \dfrac{S_r(n)}{S(n)} & (S(n) > 0) \\ 0 & (otherwise) \end{cases}$$

where,

Sr(n): IoT 소프트웨어의 교체 가능한 서비스 갯수

S(n): IoT 소프트웨어의 서비스 총 갯수

2.3. A Metric for Measuring Extensibility of IoT Software

IoT 소프트웨어에서의 확장성(Extensibility)이라 함은 IoT 소프트웨어 내의 서비스들 가운데 확장 가능한 서비스들의 비율을 의미한다[13,14].

[정의 3] 확장성(RSE: Rate of Extensibility)

RSE(n)는 IoT 소프트웨어 내의 서비스들 가운데 확장 가능한 서비스들의 비율을 의미한다.

$$RSE(n) = \begin{cases} 1 - \dfrac{S_e(n)}{S(n)} & (S(n) > 0) \\ 0 & (otherwise) \end{cases}$$

where,
Se(n): IoT 소프트웨어의 확장 가능한 서비스 갯수
S(n): IoT 소프트웨어의 서비스 총 개수

IV. Experiments and Evaluation

이 장에서는 본 논문에서 제시하는 IoT 소프트웨어의 서비스 재사용성을 측정하기 위해 변경성, 교체성, 확장성을 측정하는 메트릭을 적용한 사례연구를 제시하고, 이를 적용하여 재사용성을 측정하고자 한다. 본 논문에서 사례연구로 작성한 안드로이드 서비스 개발 환경은 하드웨어는 Intel i7, 16G RAM, 256G SSD이고, 개발 소프트웨어는 안드로이드 스튜디오 3.4.2, ADK 9.+(Q) API 29, JDK 1.8 버전을 적용하였다.

1. Experiments

1.1 변경가능성에 대한 사례 연구

Fig. 1은 서비스 내의 흐름 변경이 불가능한 관계를 보여준다. Fig. 1과 같이 설정되면 해당 Service_1에서 Service_2를 호출하게 되는 경우 Service_2 내에서의 흐름이 한가지 형태로만 고정된다. 따라서 Service_2 내에서 다른 어떤 제어 흐름 설정 변경이 이루어질 수 없다.

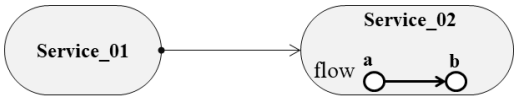


Fig. 1. No-Changeable Service Relation

이러한 관계에 대한 안드로이드[15] 코드 샘플은 Fig.2와 같이 이루어진다.

```
public class Activity_1 extends Activity {
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.caller_layout_001);
   Button myButton=(Button) findViewById(R.id.button);
   myButton.setOnClickListener(new View.OnClickListener() {
     public void onClick(View v) {
       Intent i = new Intent(IntentAndroid.this, Activity_2.class);
       startActivity(i);
     }
   });
 }
}
```

Fig. 2. Non-Changeable Service Relation's Code

Fig. 3은 Service_1에서 Service_2를 호출할 경우 Service_2 내의 흐름에 대한 설정이 변경 가능한 경우이다.

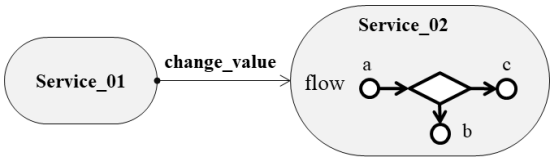


Fig. 3. Changeable Service Relations

Fig. 3과 같이 변경 가능한 관계에 있게 되면 코드는 Fig. 4와 Fig. 5에 제시된 형태로 구현된다.

```
public class Activity_1 extends Activity {
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.caller_layout_001);
   Button myButton=(Button) findViewById(R.id.button);
   myButton.setOnClickListener(new View.OnClickListener() {
     public void onClick(View v) {
       Intent i = new Intent(IntentAndroid.this, Activity_2.class);
       i.putExtra("flow_key", "flow_01");    // Change_value
       startActivity(i);
     }
   });
 }
}
```

Fig. 4. Changeable Service_1's Code

```
public class Activity_2 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.caller_layout_001);
  Button myButton = (Button) findViewById(R.id.button);
  myButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
      String key = getIntent().getStringExtra("flow_key");
      if(key.equals("flow_01")) {
       // flow 01
       } else {
        // flow 02
       }
    }
  });
 }
}
```

Fig. 5. Changeable Service_2's Code

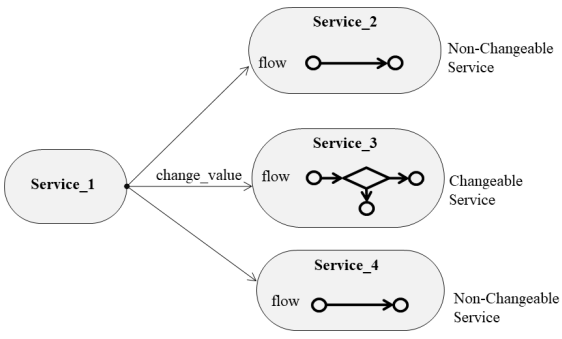


Fig. 6. Case of Non-Changeable and Changeable Services

Fig. 6과 같은 경우, 총 서비스 수는 4개이고, 변경 가능한 서비스 갯수는 Service_1과 Service_3 2개이므로 이 IoT 소프트웨어 서비스의 변경성 RSC(n)=2/4=0.5가 된다.

1.2 교체성에 대한 사례 연구

Fig. 7과 같은 사례는 교체 불가능한 서비스 관계와 교체 가능한 서비스 간의 관계를 보여준다.

Fig. 7. Non-Replaceable Service Relation

Fig. 8은 교체 불가능한 서비스 관계를 안드로이드로 구현한 코드이다.

```
public class Activity_1 extends Activity {
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.caller_layout_001);
   Button myButton=(Button) findViewById(R.id.button);
   myButton.setOnClickListener(new View.OnClickListener() {
     public void onClick(View v) {
       Intent i = new Intent(IntentAndroid.this, Activity_2.class);
       startActivity(i);
     }
   });
 }
}
```

Fig. 8. Non-Replaceable Service Relation's Code

Fig. 9는 교체 가능한 서비스 간의 관계를 표현한 것이고, Fig. 10과 Fig. 11은 이를 안드로이드로 구현한 코드이다.

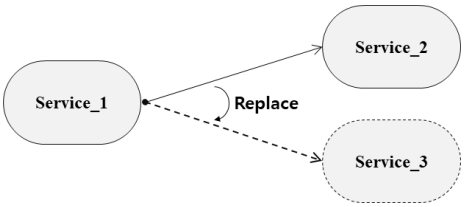


Fig. 9. Replaceable Service Relation

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton=(Button)findViewById(R.id.button);
    myButton.setOnClickListener(new
                                View.OnClickListener() {
      public void onClick(View v) {
          Intent i = new Intent();
          i.setAction(Intent.ACTION_MAIN);
          i.addCategory("_VARIABILITY_NAME");
          startActivity(i);
      }
    });
  }
}
```

_VARIABILITY_NAME is variant identifier.
(It is possible to change Activity dynamically without Activity_2.)

Fig. 10. Replaceable Service Relation's Code

```
<application android:icon="@drawable/icon" android:label="@string/activity_name1">
    <activity android:name="Activity_1"
          android:label="@string/activity_name1">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

<activity android:name="Activity_2" android:label="@string/activity_name2">
          <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.DEFAULT" />
                    <category android:name="_VARIABILITY_NAME" />
          </intent-filter>
</activity>
```

Activity_2 is changeable with Activity_3.

Fig. 11. Replaceable Configuration's Code

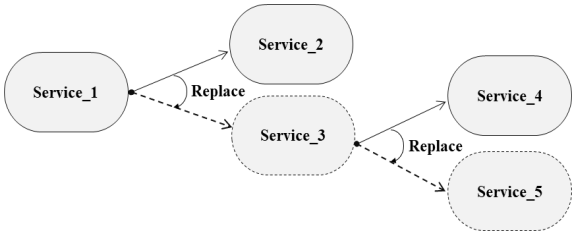


Fig. 12. Replaceable Service Relations

Fig. 12는 서비스들 간의 관계 교체 가능한 서비스로 설계된 경우로서, 이에 대한 교체성 측정 결과는 총 서비스 개수인 5개 중에 교체 가능한 서비스 수가 3개가 되므로 3/5=0.6이 된다.

1.3 확장성에 대한 사례 연구

Fig. 13은 확장 불가능한 서비스 간의 관계를 보여준다.

Fig. 13. Non-Extensible Service Relation

Fig. 13처럼 확장 불가능한 서비스들 간의 관계는 Fig. 14와 Fig. 15 같은 형태의 코드로 구현된다.

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        Intent i = new Intent(IntentAndroid.this, Activity_2.class);
        startActivity(i);
      }
    });
  }
}
```

Fig. 14. Non-Extensible Service Relation's Code

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="inha.android"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/activity_name1">
    <activity android:name="Activity_1"
          android:label="@string/activity_name1">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <activity android:name="Activity_2" > /activity>
```

Fig. 15. Non-Extensible Configuration's Code

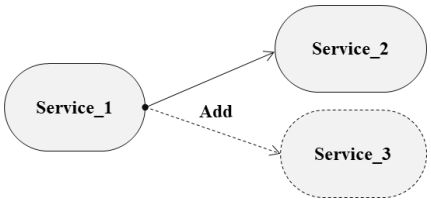


Fig. 16. Extensible Service Relation

Fig. 16은 확장이 가능한 서비스 간의 관계를 보여준다. Fig. 16처럼 확장 가능한 서비스들 간의 관계는 Fig. 17처럼 구현되고, Fig. 18과 같은 형태의 설정 코드로 구현된다.

```
public class Activity_1 extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.caller_layout_001);
    Button myButton = (Button) findViewById(R.id.button);
    myButton.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        Intent i = new Intent();
        i.setAction(Intent.ACTION_MAIN);

        i.addCategory("_VARIABILITY_NAME");
        startActivity(i);
      }
    });
  }
}
```

_VARIABILITY_NAME is variant identifier.
(It is possible to extend Activity dynamically without Activity_2.)

Fig. 17. Extensible Service Relation's Code

```
<application android:icon="@drawable/icon" android:label="@string/activity_name1">
    <activity android:name="Activity_1"
          android:label="@string/activity_name1">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

<activity android:name="Activity_2" android:label="@string/activity_name2">
        <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="_VARIABILITY_NAME" />
        </intent-filter>
</activity>
<activity android:name="Activity_3" android:label="@string/activity_name3">
        <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="_VARIABILITY_NAME" />
        </intent-filter>
</activity>
```

Fig. 18. Extensible Service's Configuration

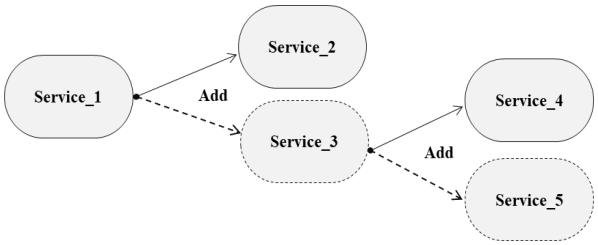


Fig. 19. Extensible Service Relations

따라서 이를 반영한 사례로 Fig. 19의 경우에는 총 서비스 수가 5개이고, 확장 가능하도록 설정된 서비스 수가 3개이기 때문에 3/5=0.6이 된다.

사례 연구에서 보여주는 것처럼 변경성, 교체성, 확장성을 합산하여 이에 대한 평균을 계산하면 해당 IoT Software 서비스의 재사용성에 대한 측정값이 계산된다. 위 경우에서는 변경성이 0.5, 교체성이 0.6, 확장성이 0.6이기 때문에 이에 대한 재사용성 값은 0.56이 된다. 재사용성 측정값은 가장 높은 경우가 1에 해당하게 된다.

V. Conclusions

본 연구를 통해 얻을 수 있는 기대효과는 다음과 같다. 첫째, IoT 소프트웨어 개발 시 향후 재사용성이 높은 서비스들을 개발할 수 있다. 둘째, 개발될 IoT 소프트웨어의 서비스에 대한 품질과 재사용성을 측정할 수 있는 도구로 사용할 수 있다. 셋째, IoT 소프트웨어의 서비스가 확장 가능한 형태로 개발할 수 있다. 본 논문에서 제시한 메트릭은 안드로이드를 적용한 IoT 소프트웨어 개발에 적용한 경우로서, IoT 소프트웨어의 서비스 개발 생산성 향상을 가져올 수 있을 뿐만 아니라 IoT 소프트웨어 서비스 개발 및 유지보수에 있어서 재사용성의 향상을 기대할 수 있다. 이는 결국 IoT 소프트웨어의 서비스 품질을 향상시킬 수 있다. 향후 연구로는 ISO/IEC 25000을 기반으로 IoT 소프트웨어의 7가지 품질 측정 항목 중 호환성, 이동성, 하드웨어와의 공존성, 실시간 접근성 등에 대한 메트릭을 추가적으로 개발하고, 제안한 IoT 소프트웨어 서비스의 측정 메트릭들을 반영하여 자동화 도구를 설계하고 개발하여 IoT 소프트웨어 서비스의 품질을 향상시키는 것이다.

ACKNOWLEDGEMENT

The present research has been conducted by the Research Grant of Seoil University.

REFERENCES

[1] Jae-Ho Kim, Jae-Seok Yoon, Sung-Chan Choi, Min-Woo Rhew, “Development Trends and Direction of Advance of IoT Platform”, Information and Communications Magazine, Vol.30, Issue 8, pp.29-39, July, 2013, DOI: https://www.koreascience.or.kr/article/JAKO201302757805681.page

[2] S. J. Kim, D. E. Cho, “Technology Trends for IOT Security”, The Korea Contends Association, Vol. 13, No. 1, pp.31-35, March, 2015, DOI: https://doi.org/10.20924/CCTHBL.2015.13.1.031

[3] Disruptive Civil Technologies, URL: http://www.fas.org/irp/nic/disruptive.pdf, April 2008.

[4] S. C. Noh, and J. G. Kim, “A Study on Matrix Model for Core Quality Measurement Based on the Structure and Function Diagnosis of IoT Networks”, Convergence Security Journal, Vol. 14, Issue 7, pp.45-51, December, 2014, DOI: http://ksci.kisti.re.kr/search/article/articleView.ksci?articleBean.atclMgntNo=SOBTCQ_2014_v14n7_45

[5] M. Kim, “A Quality Model for Evaluating IoT Applications”, International Journal of Computer and Electrical Engineering, Vol. 8, No. 1, pp.66-76, February, 2016, DOI: 10.17706/ijcee.2016.8.1.66-76

[6] M. Kim, N. Y. Lee, J. H. Park, “A Quality Evaluation Model for IoT Services”, KIPS Tr. Comp. and Comm. Sys., Vol. 5, No. 9, pp.269-274, September, 2016, DOI: https://doi.org/10.3745/KTCCS.2016.5.9.269

[7] S. H. Oh, H. J. Rha, S. D. Kim, “Method to Evaluate and Enhance Reusability of Cloud Services”, The KIPS Transactions, Part D, Vol. 19, No. 1, pp.49-62, February, 2012, DOI: https://doi.org/10.3745/KIPSTD.2012.19D.1.049

[8] Suryn W., Abran A., Bourque P., Laporte C., “Software Product Quality Practices - Quality Measurement and Evaluation using TL9000 and ISO/IEC 9126”, 10th International Workshop on Software Technology and Engineering Practice - STEP 2002, Montréal (Canada), IEEE-Computer Society Press, Los Alamitos, pp. 156-162, October 2002, DOI: 10.1109/STEP.2002.1267625

[9] I. G. Park, S. D. Kim, "Software Component Reusability Metrics", Journal of KISS: Software and Applications, Vol.31, No.6, pp.760-772, June, 2004, DOI: https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE00617662

[10] S. M. Chung, J. H. Choi, and J. W. Park, “Design of Soware Quality Evaluation Model for IoT”, Journal of the Korea Institute of Information and Communication Engineering, Vol. 20, Issue 7, pp.1342-1354, 2016, DOI: https://doi.org/10.6109/jkiice.2016.20.7.1342

[11] K. S. Kwak,, “IoT with Software-aware Issues”, Journal of Communications of the Korean Institute of Information Scientists and Engineering, Vol. 32, No. 6, pp.9-18, June 2014, DOI: http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE02433149

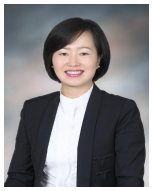
[12] John Estdale, Elli Georgiadou, “Applying the ISO/IEC 25010 Quality Models to Software Product”, European Conference on Software Process Improvement, pp. 492-503, September, 2018, DOI: 10.1007/978-3-319-97925-0_42

[13] Tommi Mikkonen, Antero Taivalsaari, "Software Reuse in the Era of Opportunistic Design", IEEE Software, Vol. 36, Issue. 3, April, 2019, DOI: 10.1109/MS.2018.2884883

[14] E. S. Cho, C. J. Kim, C. Y. Song, "Development of Metrics to Measure Reusability of Mobile App", Journal of the Korea Academia-Industrial Cooperation Society, Vol. 15, No. 7, pp.4500-4507, July, 2014, DOI: https://doi.org/10.5762/KAIS.2014.15.7.4500

[15] S. H. Kim, "Android Programming Complete Guide", Hanbit Media, April, 2013.

Authors



Eun-Sook Cho received the B.S. degree in Computer Science from DongEui University, Korea in 1993. She received the M.S and Ph.D degree in Computer Science from SoongSil University, Korea, in 1996 and 2000, respectively. Dr. Cho joined the faculty of the Department of Software Engineering at Seoil University, Seoul, Korea, in 2005. She is currently a Professor in the Department of Software Engineering, Seoil University. She is interested in framework modeling and development, Big Data, AI, Service-Oriented Computing, and IoT Software.