KIPS Trans. Softw. and Data Eng.
Vol.11, No.6 pp.229~236
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2022.11.6.229

Study for Design of Defect Management to Improve the Quality of IoT Products

Kim Jae Gyeong† · Choi Yeong Sook† · Cho Kyeong Rok†† · Lee Eun Ser†††

ABSTRACT

Based on the Internet of Things, a web system that can check the condition around the fire extinguisher, whether a fire has occurred, and an application that can receive fire notifications in real time is implemented. Minimize errors that occur during development by using software engineering to clarify the goals of the system and define the structure in detail. In addition, for IoT-based fire extinguishers, a method of reducing defects by finding product defects in the demand analysis, design, and implementation stages and analyzing the cause thereof is proposed. Through the proposed research, it is possible to secure the reliability of defect management for IoT-based smart fire extinguisher.

Keywords : IoT, Fire Detection, Defect Management, Factor Analysis, Defect Tracking

IoT 제품의 품질 개선을 위한 결함관리 설계에 관한 연구

김 재 경† · 최 영 숙† · 조 경 록†† · 이 은 서†††

요 약

사물인터넷을 기반으로 실시간으로 소화기 주변의 상태와 화재 발생 여부, 소화기의 상태를 확인할 수 있는 웹 시스템과 화재 알림을 받을 수 있는 애플리케이션을 구현한다. 해당 시스템의 목표를 명확하게 정하고 구조를 상세히 정의함으로써 개발 도중 일어나는 오류를 최소화한다. 또한, 스마트소화기에 대한 요구분석과 설계, 구현 단계에서 제품 결함을 찾아내고 그 원인을 분석하여 결함을 줄이는 방법을 제안한다. 제안하는 연구를 통해 IoT 기반의 스마트 소화기에 대해 결함관리 신뢰성을 확보할 수 있다.

키워드 : 사물인터넷, 화재 발생 감지, 결함관리, 요인분석, 결함추적성

1. 서 론

전국 곳곳에서 일어나는 화재로 많은 인명피해와 재산피해가 끊임없이 이어지고 있다. 소방청에서 제공하는 국가화재정보시스템에 의하면 2020년 동안 발생한 화재사건 중 60% 이상이 건축, 구조물에서 발생하였다. 특히 공장의 경우에는 특정 인원이 장시간 머무는 경우가 많고 건물의 규모가 비교적 크기 때문에 인적이 드문 곳에서 발생하는 화재를 인식하기 어렵다. 또한 화재를 발견했음에도 근처에 소화기를 찾을 수 없어 조기 진압이 이루어지지 않을 수도 있다. 본 연구는 주변 온도, 습도를 측정하는 라즈베리파이를 소화기에 부착함으로써 실시간으로 수치를 확인할 수 있다. 또한 화재 발생 시에 사용자에게 화재 알림을 전송함으로써 화재를 빠르게 인지하고 즉각 대응하도록 유도한다. 본 연구를 통하여 화재 진압 도구인 소화기와 사물인터넷을 결합하여 화재를 초기에 발견하고 건물 내부의 사람들에게 화재 발생 공지 및 대피 명령을 신속히 전달하는 시스템을 구축하여 화재로 인한 재산피해 및 인명피해를 최소화할 수 있을 것으로 예상된다. 한편, IoT 기반의 소화기에 대해 요구분석과 설계 및 구현 단계 등 프로세스 관리 측면에서 어떤 결함이 있는지 찾아내어 그 결함원인을 분석하고 검출된 결함을 줄이는 방법을 연구하고자 한다.

2. 기반 연구

2.1 요인분석

요인분석이란 현실의 자료에서 요인을 추출하는 방법이다. 요인분석은 목적에 따라 두 가지로 나뉜다. 그 중 탐색적 요인분석은 분석의 대상이 되는 많은 측정 요인들의 변수들을 상관관계가 높은 것끼리 묶어서 변수를 단순화 시키는데 사

※ 이 논문은 2021년 한국정보처리학회 ACK 2021의 우수논문으로 "IoT기반 스마트 소화기 구현"의 제목으로 발표된 논문을 확장한 것임.
† 비 회 원 : 안동대학교 컴퓨터공학과 학사과정
†† 준 회 원 : 한국소비자원 전기전자팀 연구위원
††† 종신회원 : 안동대학교 컴퓨터공학과 교수
Manuscript Received : December 28, 2021
Accepted : January 12, 2022
* Corresponding Author : Lee Eun Ser(eslee@anu.ac.kr)

용하는 것이다. 즉, 수많은 변수들 중에서 잠재된 몇 개의 변수(요인, 문항)를 찾아내는 것이다[1,2]. 한편 요인모형에 대한 특정한 가정이나 기존 연구에서의 요인모형을 경험적인 데이터를 사용하여 검증하는 확인적 요인분석이 있다[3].

2.2 결함분석

결함은 소프트웨어의 개발 생명주기인 요구사항분석, 설계, 구현, 유지보수 등 모든 단계에서 발생할 수 있다. 그러나 이러한 결함은 신뢰성을 떨어뜨리고 잠재적인 문제를 발생해서 커다란 파장을 발생시키게 된다. 따라서 발생한 결함의 원인을 해결해야만 신뢰성 있는 소프트웨어가 될 수 있다[4,5].

2.3 결함추적성 연관성 분석

결함분석을 위해서는 결함의 요인이 되는 항목들에 대해서로 연관성이 있는 항목들이 무엇인지 찾는 것에서부터 출발한다고 본다. 이를 위해서는 “요구사항의 추적성을 양방향으로 유지하는 것이며, 요구사항이 잘 관리되는 경우는 요구사항의 원천으로부터 하위 수준의 요구사항으로, 그리고 하위 수준의 요구사항으로부터 그 원천으로의 추적성이 확립될 수 있다”[6-8].

3. 연구 방법

3.1 요구 분석

요구 분석이란 새로 만들어질 시스템 또는 변경해야 하는 시스템의 요구를 정의하는 작업이다. 요구 분석을 할 때 제일 처음에 수행해야 하는 것은 현재 시스템의 상태를 파악하고 요구를 정의한다. 그리고 구현해야 하는 시스템의 목표를 정확히 도출한다. 다음 단계에서는 구현할 소프트웨어가 어떤 기능을 포함하는 지에 대해 정확히 기술해야 한다. 요구사항 명세서에 이러한 사항들을 기술하는데, 기술적 요구와 구현의 제약조건과 성능에 관하여 사용자와 개발자 사이의 일어난 합의 사항도 명시되어야 한다. [9] 본 연구에서는 기능의 상세설명과 중요도, 난이도를 나타내는 표를 제시한다. 이때 중요도는 상, 중, 하로 나타내며 중요도가 높을수록 해당 시스템이 본래의 목표를 이행하는 데 크게 작용한다. 중요도가 ‘상’이라는 것은 목표 시스템에서 해당 기능이 절대적으로 포함되어야 한다. 중요도가 ‘중’이라는 것은 해당 기능의 구현이 권고되지만 필수적이지는 않다. 난이도가 ‘하’라는 것은 요구로 인하여 도출된 기능이지만 제외될 가능성이 있다. 난이도 역시 상, 중, 하로 나타내었고, 난이도가 ‘상’이라는 것은 해당 요구가 여러 요구와 맞물려 있고 데이터 통신을 하고 있는 경우이다. 난이도가 ‘중’이라는 것은 데이터 통신을 하여 어떠한 정보를 송, 수신하는 경우이다. 난이도가 ‘하’라는 것은 해당 요구가 가장 작은 단위의 요구이며 다른 요구와 독립적인 경우이다.

3.2 결함 분석

각 단계별 문제점에 대한 원인 분석을 하고 그 해결책 방안을 위해 결함관리가 필요한 것으로 사료된다. 따라서 요구사항의 문제점에 대한 결함관리를 위해 조사·분석하였다[8,9].

본 논문에서는 요인 분석을 이용하여 생명주기의 한 단계에서 해당 결함이 다른 결함과 연관성이 있는지 파악하고 각 생명주기마다 결함에 대한 영향도가 가장 높은 요인을 선정하여 생명주기의 한 사이클에서 발생할 수 있는 결함점수를 계산한다. 결함 점수를 계산하는 계산식은 아래와 같다.

결함점수(C) = 연관결함개수(A) + (연관기능개수(B)*0.5) (1)

이때, 결함 추적표를 이용하여 해당 결함과 연관되는 기능의 개수를 알아낸다. 결함 추적표란 특정 결함이 설계, 요구사항의 어떤 단계에 영향을 미치고 있는지 정리한 표이다. 연관결함개수(A)에 연관기능개수(B)*0.5를 더하는 이유는 연관결함개수(A)가 하나도 없을지라도 이 결함에 대한 영향도를 무시해서는 안 되기 때문이다. 그렇기 때문에 연관결함개수(A)가 없는 경우에도 연관기능개수(B)를 계산하여 결함의 영향도를 측정한다. 단, 연관결함개수(A)는 다른 결함과 연관성이 있는 반면에 연관기능개수(B)는 해당 결함이 제거되면 연관된 기능이 정상 작동할 가능성이 있으니 영향도는 연관결함개수(A)보다 작게 측정한다.

해당 내용을 Fig. 1을 참고하여 정리해보자면 요구사항 분석 단계에서 요인1과 요인2가 도출되었을 때, 요인1의 항목들에 대하여 요인1과 요인2 항목들의 연관성과 연관결함개수(A), 연관기능개수(B)를 조사한다. 이때 위의 계산식을 통하여 결함점수를 도출해낸다. 요인2에도 동일한 수행을 적용하여 둘 중 더 높은 결함점수(결함합계점수(D))를 가진 요인을 요구

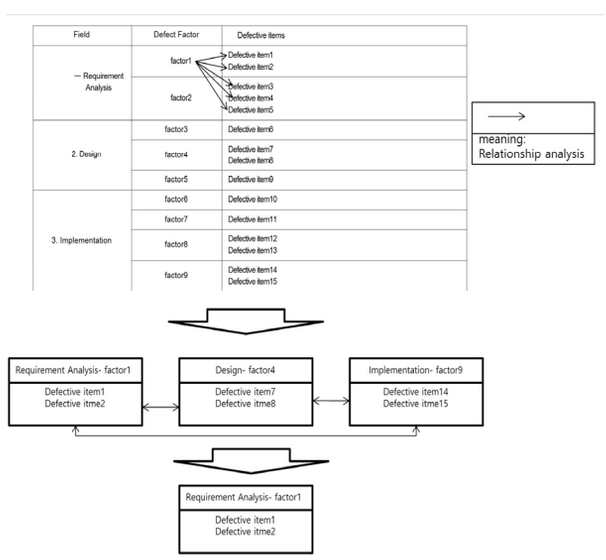


Fig. 1. Defect Analysis

분석의 대표 요인으로 정한다. 설계, 구현에서도 같은 작업을 수행하며 설계와 구현에서도 각 단계의 대표 요인이 하나씩 도출된다.

이러한 과정을 거쳐 최종적으로 결함의 영향도가 가장 높을 것이라 예상되는 요인을 도출하여 해당 결함요인과 전체 결함에 대한 연관성을 분석한다. 결함분석을 통해서 어떤 영역의 요인이 전체 결함에 가장 큰 영향을 주고 있는지 파악할 수 있고, 해당 결함요인을 제거하면 해당 결함요인과 연관성이 있는 결함이 제거될 가능성이 있다. 최대 결함요인과 관련된 결함 개수/전체 결함 개수(%) 만큼의 결함을 줄일 수 있다.

4. 사례 연구

4.1 환경 설정

하드웨어와 소프트웨어의 제원은 다음과 같다.

- CPU : Intel(R) Core(TM) i7-8750u, 2.20㎓,
- RAM : 8.0GB
- 시스템 종류 : 64비트운영체제, X64 기반프로세서
- 디스플레이 어댑터 : Intel(R) HD Graphics 620
- 소프트웨어 : Windows 10pro

4.2 요구사항 분석

요구사항 분석은 소프트웨어 개발 생명주기의 첫 단계이다. 아래의 〈Table 1~3〉은 본 연구에서 연구한 '3.1 요구분석'을 기반으로 작성하였다. 〈Table 1〉에는 요구되는 기능과 기능에 대한 간략한 설명, 중요도, 난이도가 서술되어 있다. 〈Table 2〉는 웹페이지에 온습도 값 출력 기능의 요구사항 명세서이다. 해당 요구의 개요, 세부사항, 유형, 중요도, 우선순위 등이 표시된다. 〈Fig. 2〉은 요구사항 명세서를 기반으로 작성한 유스케이스 다이어그램이다. 유스케이스 행위자와 기능 간의 관계를 구조적으로 나타낸다. 〈Table 3〉은 〈Table 2〉

Table 1. Requirements Definition

Type	Detailed explanation	Importance	Difficulty
Function	Measure the temperature and humidity	Major	Minor
	Output a warining sound	Major	Minor
	Detect a fire	Major	Medium
	Output LED lamp	Major	Minor
	Measure the angle	Medium	Medium
	Join	Major	Medium
	Login	Major	Medium
	Fire alarm	Major	Major
	Check if there's a fire	Major	Major
	Check the temperature and humidity value	Major	Major
	Check the angle	Minor	Major

Table 2. Requirements Specification

Requirements		Display the temperature and humidity on web page		
summery		The temperature and humidity values stored in the DB server are transferred to the web server.		
Contents	Detail	Data is received from the DB server in which the temperature and humidity value measured by the raspberry pie is stored. And Display the temperature humidity value on the web page.		
	Type	Function		
	Importance	Major	Difficulty	Major

Table 3. Usecase Specification

Usecase name	Temperature and humidity value notification
Actor	Web Server Manager
Precedenc conditions	External DB server and web server can communicate.
Precedenc input	Temperature and humidity values stored in the external DB server are transmitted to the web server.
Event flow	1. The temperature and humidity values stored in the external DB server are transmitted to the web server. 2. Output the delivered temperature and humidity values to the web server. 3. The administrator monitors the temperature and humidity values through a web server.
Trailing conditions	
Trailing output	The administrator monitors the temperature and humidity values through a web server.
Restrictions	

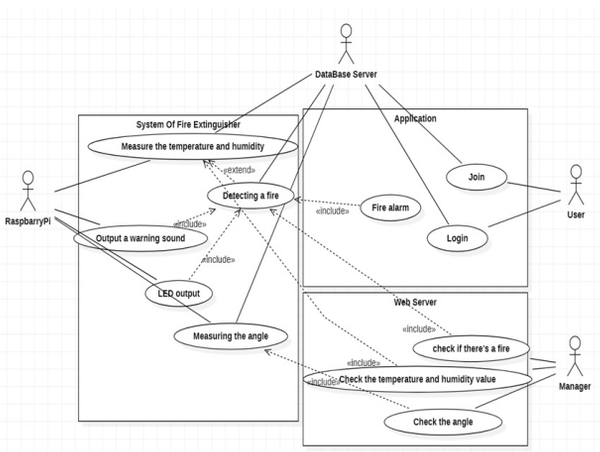


Fig. 2. Usecase Diagram

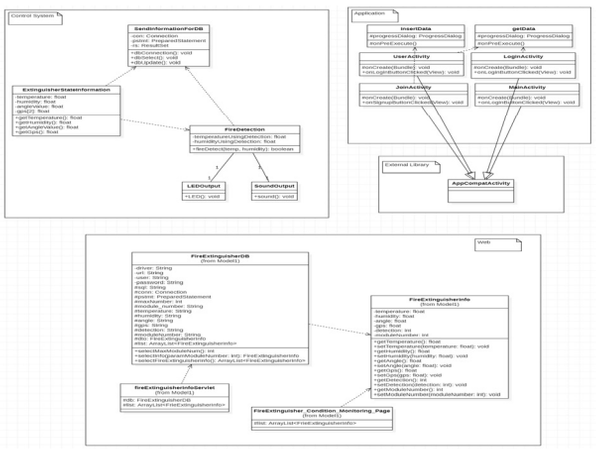


Fig. 3. Class Diagram

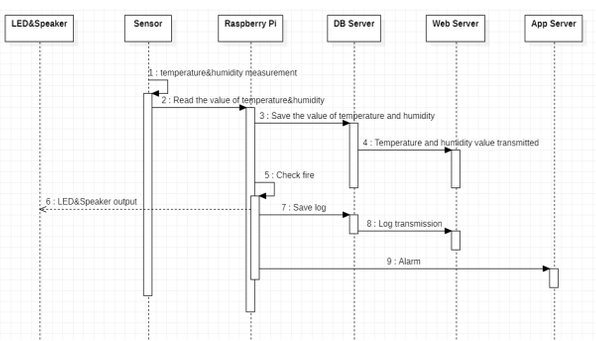


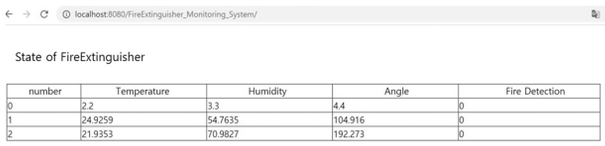
Fig. 4. Sequence Diagram

의 유스케이스 명세서이다. 해당 요구의 유스케이스명, 액터, 선행조건, 선행입력, 이벤트흐름, 후행조건, 후행조건, 제약사항 등을 서술한다. 〈Table 2〉, 〈Table 3〉가 서술하는 기능은 '온습도 값 알림' 이며, 라즈베리파이와 DB서버, DB서버와 웹서버가 통신이 가능할 때 라즈베리파이가 외부DB에 저장해 둔 온습도 값을 웹서버로 전달받는다. 전달된 온습도 값을 웹서버에 출력하면 관리자가 웹페이지를 통해 온습도 값을 모니터링한다.

4.3 설계

설계에서는 UML기법 중 하나인 클래스 다이어그램을 사용하였다. 클래스 다이어그램은 각 클래스는 자신의 특성을 나타내는 속성들과 오퍼레이션들로 구성된다. 유스케이스 시나리오를 분석하여 클래스의 속성, 오퍼레이션 등을 추출하고 클래스 간의 연관성을 표시한다. 〈Fig. 3〉은 해당 시스템에 대한 클래스 다이어그램을 작성한 것이다. 클래스 다이어그램은 유스케이스 정적분석에 의하여 만들어진다. 반대로 시퀀스 다이어그램은 유스케이스 동적 수행 방법이다. 시스템을 구성하는 객체들의 상호작용, 객체 상태, 동작 등을 분석하여 클래스들의 오퍼레이션을 추출한다. 〈Fig. 4〉는 해당 시스템에 대한 시퀀스 다이어그램을 작성한 것이다.

제어시스템 부분에서 ExtinguisherStateInformation클래스와 SendInformation클래스, FireDetection클래스는 의존관계이다. 정보를 보내고, 해당 정보에 대해서 화재임을 판단하기 위해서는 소화기 상태 정보 클래스로부터 값을 받아야 하기 때문이다. FireDetection 클래스와 SendInformation ForDB도 의존관계이며 이는 화재 감지 판단 값을 데이터베이스에 저장하기 위함이다. FireDetection클래스와 LEDOutput, SoundOutput은 연관관계를 가진다. 화재를 감지하는 즉시 LED와 사운드를 출력하기 위함이다. 웹 설계에서 FireExtinguisherDB와 fireExtinguisherInfoServlet이 의존관계이다. DB로부터 받은 정보를 Servlet에서 가공하기 위함이다. FireExtinguisehrDB는 FireExtinguisehrInfo와도 의존관계를 형성하는데, 이는 데이터베이스에서 읽어온 정보 값들을 FireExtinguisehrInfo의 인스턴스 형태로 관리하기 위함이다. FireExtinguisher_Condition_Monitoring_Page는 FireExtinguisherInfo와 의존관계인데, 이는 모니터링 페이지에서 FireExtinguisherInfo 객체를 전달받아 웹페이지에 표시하기 때문이다. 애플리케이션에서 InsertData JoinActivity는 의존관계이다. 데이터를 DB에 저장하는 InsertData를 통해 JoinActivity가 수행된다. getData와 LoginActivity도 의존관계이다. 마찬가지로 LoginActivity를 수행하기 위해서는 데이터를 가져오는 getData 인스턴스가 필요하기 때문이다. getData와 UserActivity는 의존관계인데, UserActivity에서 화재 감지 값을 일정 시간마다 읽어오기 때문에 데이터를 가져오는 행위를 하는 getData 인스턴스가 필요하다.



number	Temperature	Humidity	Angle	Fire Detection
0	2.2	3.3	4.4	0
1	24.9259	54.7635	104.916	0
2	21.9353	70.9827	192.273	0

Fig. 5. FireExtinguisher Monitering Page

Fig. 6. Application Beginning Screen

4.4 구현

본 논문에서는 〈4.2 요구사항 분석〉 과 〈4.3 설계〉를 이용하여 IoT(Internet Of Things) 기반의 스마트소화기를 구현하였다. 라즈베리파이가 온습도 값과 위치를 측정하여 데이터베이스에 저장하고, 해당 정보를 〈Fig. 5〉의 웹페이지로 출

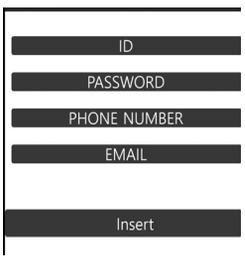


Fig. 7. User Application Join Screen

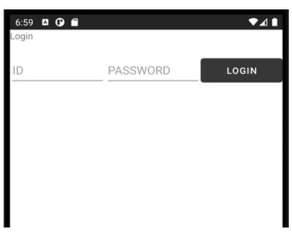


Fig. 8. User Application Login Screen

Fig. 9. User Application Main Screen

력하여 관리자가 모니터링 할 수 있게 한다. 또한 소화기의 각도도 같은 방법으로 웹페이지에 출력하여 소화기의 쓰러짐 상태를 모니터링 할 수 있다. 또한 라즈베리파이가 읽은 온습도 값을 화재라 판단했을 시 이를 DB(DataBase)에 저장하여 웹페이지가 화재여부를 유의미한 값으로 표시하도록 하고, LED와 경고음을 출력하여 주변 사람들에게 화재의 발생 및 소화기의 위치를 인지하도록 한다. 〈Fig. 6〉는 화재 발생 시 알람을 받을 수 있는 애플리케이션의 초기 화면이다. 〈Fig. 7〉을 통해 전용 애플리케이션에 가입하여 〈Fig. 8〉의 로그인을 수행하고 〈Fig. 9〉인 메인화면에 진입해 있는 사용자에게는 화재의 발생과 화재 발생을 감지한 소화기 위치를 〈Fig. 10〉처럼 알람 형태로 출력하여 신속한 대피를 유도한다.

4.5 결함분석

IoT 기반의 소화기 구현을 위한 요구사항 분석 중 요구분석, 설계, 구현 단계에서 〈Table 4〉와 같이 문제점이 도출되었다. 분석 방법은 〈Table 4〉의 영역별로 진행하면 된다.

첫째, 영역 1.요구분석 중에서 결함영역 1.1 인터페이스의 세부결함항목 1.1.1부터 1.2.6까지 13개 사이에 결함항목이 서로 연관이 있는지를 조사하여 각 세부결함 항목별로 연관

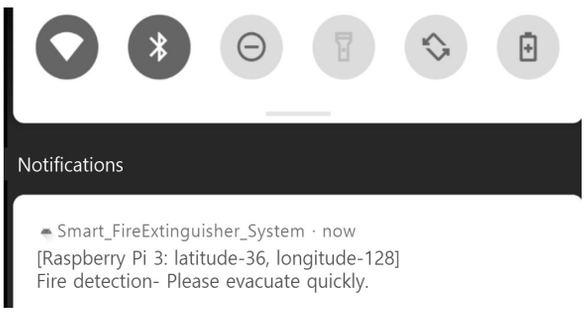


Fig. 10. Fire Detection Alarm

된 개수를 확인한다.

둘째, 각각의 세부결함항목별(1.1.1~1.1.7)로 기능을 수행하는데 연관된 기능(명세서 상에 있는 기능들)이 어떤 것이 있는지를 조사하여 연관기능개수(B)를 구한다.

셋째, 각각의 세부결함항목별로 첫째에서 구한 연관결함개수(A)와 두 번째에서 구한 연관기능개수(B)*0.5를 더해서 결함점수(C)를 구하면 된다.

넷째, 각각의 결함항목별로 구한 결함점수(C)를 모두 더하면 결함합계점수(D)가 된다. 이때 결함합계점수(D)는 1.요구분석 영역 중 결함영역 1.1 인터페이스의 결함합계점수가 된다. 이와 같이 결함영역 1.2 내부구조에 대해서도 위의 첫째부터 네 번째 방법과 같이 하여 결함합계점수(D)를 구한다.

다섯째, 1.1 인터페이스 결함합계점수와 1.2 내부구조 결함합계점수를 더하여 영역결함점수(E)를 구한다. 따라서 구해진 영역결함점수(E)가 영역 1.요구분석의 최종 결합점수가 되는 것이다. 이와 같이 2.설계와 3.구현도 구하여 영역결함점수를 구하면 된다. 예를 들어,'1.1.5 라즈베리파이 모듈 식별에 대한 정보 없음'결함이 '1.1.1 웹페이지에서 화재유무 확인 불가능', '1.1.3 소화기 위치 값을 알 수 없음', '1.1.4 소화기 각도 변경 값을 알 수 없음', '1.2.6 화재알림 제약사항 중 복수 개 응답에 대한 사항 없음' 총 5개 결함과 연관이 있다면 연관개수(A)는 5가 된다. 그리고 '1.1.5 라즈베리파이 모듈 식별에 대한 정보 없음' 결함과 연관된 기능이 4개라면 기능연관개수(B)는 4가 된다. 따라서 '1.1.5 라즈베리파이 모듈 식별에 대한 정보 없음'의 결함점수(C)는 7이 된다.

4.6 결함분석 사례

IoT 기반의 소화기 구현을 위한 요구사항 분석 중 요구분석, 설계, 구현 단계에서 발생한 문제점에 대한 요인분석 결과 〈Table 5〉와 같이 분석되었다. 각 생명주기에서 결함관리 측면에서 가장 점수가 높은 요인들을 선별하여 2차 결함분석을 수행한다. 해당 시스템의 경우 요구분석에서 '1.1 인터페이스' 와 설계에서 '2.4 데이터베이스', 구현에서 '3.3 내부로직'이 선별되었다. 2차 결함분석을 수행하여 최종적으로 산출된 결함요인은 '1.1 인터페이스'이다. 해당 요인과 전체 결함간의 연관성은 총 68개였으며, 전체 연관성 중 중복되는 항목을 제거하여 29개의 결함 항목을 추출하였다. '1.1 인터

Table 4. Defect Analysis Table

Field	Defect Factors	Defective items
1. Requirements	1.1 Interface	1.1.1 Unable to check whether there is a fire on the web page. 1.1.2 Fire extinguisher position correction function exists on the web page. 1.1.3 Fire extinguisher position value unknown 1.1.4 Fire extinguisher angle change value is unknown. 1.1.5 No information on RaspberryPi module identification. 1.1.6 No input conditions when logging in. 1.1.7 No entry conditions when signing up for membership.
	1.2 Internal structure	1.2.1 Unable to specify who will be notified of the fire. 1.2.2 Determining whether a fire is detected is performed with a value of old data rather than recent data. 1.2.3 No output restrictions for warning sound and LED output scenario. 1.2.4 Application can't communicate with RaspberryPi. 1.2.5 None of the fire notification restrictions regarding multiple responses 1.2.6 Measurement of Temperature and Humidity, No matters regarding the measurement cycle in the fire extinguisher angle measurement scenario.
2. Design	2.2 Internal structure	2.2.1 No function for password encryption. 2.2.2 No function to calculate the fire extinguisher angle change value 2.2.3 No ID verification function when logging in. 2.2.4 No password verification function when logging in. 2.2.5 No ID verification function when registering as a member. 2.2.6 No password verification function when registering as a member.
	2.3 Internal Logic	2.3.1 Unable to select data according to module number in web page design. 2.3.2 Unable to select the latest data 2.3.3 A plurality of RaspberryPi detection values cannot be displayed on the web page.
	2.4 Database	2.4.1 RaspberryPi detection value is stored in DB, and there is no RaspberryPi identifier. 2.4.2 No designator for when the RaspberryPi detection value was stored in the DB. 2.4.3 Limitation of expression according to data type. 2.4.4 Inapplication Design, Android Studio's own policy prevents direct access to external DB. 2.4.5 Primary Key Not Specified
3. Implementa-tion	3.1 Interface	3.1.1 Unable to check the fire extinguisher angle change value on the web. 3.1.2 Only one notification can be processed in the application. 3.1.3 In the event of a fire, it is impossible to notify which RaspberryPi detected it. 3.1.4 Only information about a single RaspberryPi can be displayed.
	3.3 Internal Logic	3.3.1 After measuring the fire extinguisher angle, the value of change should be shown. However, it shows the current fire extinguisher angle. 3.3.2 Inappropriate cycle reading from the sensor 3.3.3 Fire detection algorithm inaccurate 3.3.4 Angle change determination algorithm inaccurate 3.3.5 No duplicate IDs when signing up for membership. 3.3.6 When logging in, you can log in with multiple passwords with the same ID. 3.3.7 When logging in, code injection is possible by inserting and executing a specific string into the ID field. 3.3.8 When logging in, code injection is possible by inserting a specific string into the password field and executing it. 3.3.9 Code injection is possible by inserting and executing a specific string into the ID field when signing up for membership. 3.3.10 Code injection is possible by inserting and executing a specific string into the password field when signing up for membership. 3.3.11 When importing data from a database, only a single RaspberryPi measurement information can be imported. 3.3.12 When the fire detection algorithm determines the current data value as a fire, it puts the LED output and warning sound in an infinite loop to execute it indefinitely. 3.3.13 Data read from database is not the latest data.
	3.3 Database	3.4.1 External DB Direct Access Inability in Application Implementation 3.4.2 DB Connector version does not fit mysql version. 3.4.3 Library is not exported when linking databases 3.4.4 No identifier column for RaspberryPi module in database table for RaspberryPi measurement values 3.4.5 No information column on when the measured value is stored in the database table for the RaspberryPi measured value. 3.3.6 No identification key in the member database table
	3.4 Hardware	3.5.1 The temperature is measured higher than the actual temperature due to the heat generation of RaspberryPi. 3.5.2 Lack of GPIO to attach sanshat, LED, and buzzer. 3.5.3 No Switch

Table 5. The Result of the Total Score of Defective Items

Field	Defect Factors					
1. Requirements	1.1 Interface	The number of defects associated (A)	The number of related functions (B)	Defect Score (C)	Total Defect Score (D)	Field Defect Score (E)
1.2 Internal structure	1.1 Interface	1.1.1: 3 1.1.2: 2 1.1.3: 2 1.1.4: 2 1.1.5: 5 1.1.6: 3 1.1.7: 3	1.1.1: 1 1.1.2: 1 1.1.3: 1 1.1.4: 2 1.1.5: 4 1.1.6: 2 1.1.7: 2	1.1.1: 3.5 1.1.2: 2.5 1.1.3: 2.5 1.1.4: 3 1.1.5: 7 1.1.6: 4 1.1.7: 4	25	40.5
	1.2 Interner structure	1.2.1: 3 1.2.2: 2 1.2.3: 1 1.2.4: 3 1.2.5: 2 1.2.6: 2	1.2.1: 1 1.2.2: 3 1.2.3: 2 1.2.4: 3 1.2.5: 1 1.2.6: 3	1.2.1: 3.5 1.2.2: 3.5 1.2.3: 2 1.2.4: 4.5 1.2.5: 2.5 1.2.6: 3.5	20.5	
2. Design	2.2 Internal Structure	2.2.1: 0 2.2.2: 0 2.2.3: 1 2.2.4: 1 2.2.5: 1 2.2.6: 1	2.2.1: 2 2.2.2: 1 2.2.3: 2 2.2.4: 2 2.2.5: 2 2.2.6: 2	2.2.1: 1 2.2.2: 0.5 2.2.3: 2 2.2.4: 2 2.2.5: 2 2.2.6: 2	9.5	31.5
	2.3 Internal Logic	2.3.1: 1 2.3.2: 2 2.3.3: 1	2.3.1: 4 2.3.2: 3 2.3.3: 4	2.3.1: 3 2.3.2: 3.5 2.3.3: 3	10.5	
	2.4 Database	2.4.1: 2 2.4.2: 1 2.4.3: 1 2.4.4: 0 2.4.5: 0	2.4.1: 4 2.4.2: 3 2.4.3: 3 2.4.4: 3 2.4.5: 2	2.4.1: 4 2.4.2: 2.5 2.4.3: 2.5 2.4.4: 1.5 2.4.5: 1	11.5	
3. Implementation	3.1 Interface	3.1.1: 2 3.1.2: 2 3.1.3: 2 3.1.4: 2	3.1.1: 1 3.1.2: 1 3.1.3: 1 3.1.4: 3	3.1.1: 2.5 3.1.2: 2.5 3.1.3: 2.5 3.1.4: 3.5	11	58
	3.3 Internal Logic	3.3.1: 2 3.3.2: 2 3.3.3: 1 3.3.4: 2 3.3.5: 2 3.3.6: 2 3.3.7: 0 3.3.8: 0 3.3.9: 0 3.3.10:0 3.3.11:3 3.3.12:1	3.3.1: 1 3.3.2: 1 3.3.3: 2 3.3.4: 1 3.3.5: 2 3.3.6: 1 3.3.7: 1 3.3.8: 1 3.3.9: 1 3.3.10:1 3.3.11:4 3.3.12:4	3.3.1: 2.5 3.3.2: 2.5 3.3.3: 2 3.3.4: 2.5 3.3.5: 3 3.3.6: 2.5 3.3.7: 0.5 3.3.8: 0.5 3.3.9: 0.5 3.3.10:0.5 3.3.11:5 3.3.12:3	25	
	3.3 Database	3.4.1: 0 3.4.2: 0 3.4.3: 0 3.4.4: 3 3.4.5: 1 3.4.6: 2	3.4.1: 3 3.4.2: 4 3.4.3: 3 3.4.4: 4 3.4.5: 4 3.4.6: 2	3.4.1: 1.5 3.4.2: 2 3.4.3: 1.5 3.4.4: 5 3.4.5: 3 3.4.6: 3	16	
	3.4 Hardware	3.5.1: 0 3.5.2: 0 3.5.3: 1	3.5.1: 4 3.5.2: 4 3.5.3: 2	3.5.1: 2 3.5.2: 2 3.5.3: 2	6	

페이스'의 결함 항목을 모두 제거할 수 있다면 29('1.1 인터페이스' 요인이 영향을 주는 결함 개수)/53(전체 결함 개수), 즉 최대 54.71%의 결함을 줄일 수 있다. 이를 통하여 전체 결함에 가장 큰 영향을 주는 영역과 요인은 요구분석의 인터페이스임을 알 수 있다.

5. 결 론

본 논문에서는 사물인터넷을 기반으로 소화기 상태를 실시간으로 확인하여 웹페이지에 표시한다. 소화기의 각도를 측정하여 소화기 쓰러짐 유무를 측정할 수 있고 라즈베리파이

의 온습도, 화재발생유무, 각도, 위치 값을 관리자가 모니터링 하여 화재 발생을 빠르게 파악하도록 하며, 화재 발생 장소와 화재 발생 시간을 유추할 수 있도록 한다. 또한 사용자에게 화재 알림을 보내는 애플리케이션을 구현함으로써 해당 시스템이 설치되어있는 건물 내부의 사람들에게 화재 발생을 알리고 신속히 대피할 수 있도록 한다. 이는 화재로 인한 인명피해를 줄이는 데 일조할 것이라 예상된다. 향후에는 애플리케이션에 해당 시스템이 설치된 건물의 지도를 표시하고 사용자 위치와 라즈베리파이 위치를 추적하여 화재가 발생했을 때 안전하고 빠르게 대피할 수 있는 대피 경로를 제공할 예정이다.

한편, 요구사항 분석은 소프트웨어 개발 생명주기의 첫 단계로서 매우 중요하다고 할 수 있다. '4.6 결함분석 사례'에서 보듯이 요구분석의 요인이 결함에 가장 큰 영향을 주는 것으로 분석된 결과를 보면 사전에 결함관리가 반드시 필요하다는 것을 보여주고 있는 것으로 사료된다. 향후 연구에서는 결함관리를 위한 알고리즘 개발과 체계화된 연구를 진행하고자 한다.

References

[1] G. J. Song, "Statistical analysis of SPSS/AMOS required for preparation of revised and augmented paper," 21cbook, Gyeonggi-do, 2019.

[2] H. J. No and Xinshengxyz, "Survey and Statistics," Gyeonggi: Hakhyunsa, 2016.

[3] S. M. Lee, "The basics of factor analysis," 2000.

[4] E. S. Lee, "A study on the methodology for defect managementin the requirements stage," *Journal of the Korean Society for Information Processing*, Vol.9, No.7, pp.205-212, 2020.

[5] J. W. Jang, "Study of the improvement measurement of test project through software defect trend analysis," *Journal of the Korea Academia-Industrial Cooperation Society*, Vol.16, No.1, pp.691-696, 2015.

[6] Y. S. Jeong, "Analysis of defect reduction effects of the requirements traceability management process," 2007.

[7] M. J. Lee, "CMMI for development manual," Seoul: Hanteemedia, 2013.

[8] Information and Communication Industry Promotion Agency, "SW process training centered on SP certification,"[Internet], www.sw-eng.kr

[9] E. M. Choi, software engineering, pp.158-160, 2011.

김 재 경

https://orcid.org/0000-0003-0076-7642

e-mail : dongji_11@naver.com

2019년 ~ 현 재 안동대학교 컴퓨터공학과 학사과정

관심분야 : Software Engineering, Defect Analysis

최 영 숙

https://orcid.org/0000-0002-8276-5917

e-mail : young_s52@naver.com

2020년 ~ 현 재 안동대학교 컴퓨터공학과 학사과정

관심분야 : Software Engineering, DataBase

조 경 록

https://orcid.org/0000-0002-2174-5525

e-mail : chokr0216@naver.com

2009년 한국방송통신대학교 정보통신학과 (석사)

2018년 ~ 현 재 안동대학교 컴퓨터공학과 박사과정

1988년 ~ 현 재 한국소비자원 전기전자팀 연구위원

관심분야 : Defect Management, Software Process, Statistical Analysis

이 은 서

https://orcid.org/0000-0002-7637-3036

e-mail : eslee@anu.ac.kr

2001년 ~ 현 재 ISO/IEC 15504 국제 선임심사원

2004년 중앙대학교 컴퓨터공학과(박사)

2008년 ~ 현 재 안동대학교 컴퓨터공학과 교수

관심분야 : CBD, Formal method, Quality model, SPI(Defect Analysis)