KIPS Tr. Software and Data Eng.
Vol.4, No.1 pp.31~36 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.1.31

Study for Remove of Uncertainty by Identification of Ambiguity

Eun-Ser Lee†

ABSTRACT

There are many uncertainty items when we're working on a software. Especially, if we don't have experience in similar field, ambiguity items have a strong influence on the system entirely. Management of ambiguity items such like uncertainty things is important the factor for reliability of software. Therefore, this research is processing the evaluation criteria for remove of uncertainty items by identify of ambiguity items. Also, this research provides criteria of uncertainty identify and processing of uncertainty items, quantitative evaluation criteria to the remove of ambiguity on software development.

Keywords : Management of Uncertainty, Risk Management, Management of Reliability

모호성 식별에 의한 불확실성 제거에 관한 연구

이 은 서†

요 약

많은 불확실성 항목들은 소프트웨어 실행 시에 존재한다. 특히, 유사한 분야에서 전반적인 시스템에 많은 영향을 주게 된다. 불확실성 요소와 같은 모호성 요소의 관리는 소프트웨어 신뢰성을 위하여 중요한 요소가 된다. 따라서 모호성을 식별하여 불확실성 요소를 제거하도록 판단기준을 프로세스화 하였다. 또한 본 연구에서는 소프트웨어 개발 시, 모호성을 제거하기 위한 불확실성 식별 기준과 불확실성 처리 프로세스 및 정량적인 평가에 대하여 제시한다.

키워드 : 불확실성 관리, 위험관리, 신뢰성 관리

1. 서 론

현재의 시스템들은 장비 및 기계의 제어 시스템으로부터 정보 시스템 혹은 전자상거래 시스템에 이르기까지 다양한 종류의 컴퓨터 기반의 중대한 시스템이 존재한다. 그러한 시스템 개발에 고급 소프트웨어 공학 기술이 사용되어서 시스템의 신뢰도를 제공하고 있다. 따라서 운영되는 응용 도메인에 대한 제약사항과 특징을 알아야 하기 때문에 이해하는 것은 매우 어려울 수도 있다[13].

요구사항으로부터 개발되는 소프트웨어들은 요구사항을 만족시켜서 신뢰성을 얻으려고 한다. 이와 같은 소프트웨어 신뢰성과 밀접한 관계가 있는 요소는 불확실성이 있다. 불확실성은 소프트웨어 개발에 있어서 위험 요소가 될 수도 있으며 품질과 신뢰성 측면에서도 만족도를 저해시킬 수 있다. 따라서 불확실성에 대한 관리는 소프트웨어 개발에 있어서 중요한 요인이 된다. 이와 같은 목적을 달성하기 위하여 소프트웨어에 대한 품질 검토활동을 수행하고 있다. 소프트웨어 검토는 분석, 설계 및 코딩이라 불러온 소프트웨어 엔지니어링 활동을 정화시킨다. 검토는 소프트웨어 개발을 하는 동안 다양한 지점에 적용되며 오류들과 결함들을 발견하는 일을 수행한다[1].

소프트웨어 프로세스 관점에 소프트웨어가 실사용자에게 배포된 후에 발견된 품질 문제를 의미하는 것이 결함 요소가 된다. 오류도 소프트웨어가 실사용자에게 배포되기 전에 소프트웨어 엔지니어에 의해 발견된 품질 문제를 의미한다. 개발 소프트웨어가 품질이 저해되는 것을 방지하기 위하여 각 사항들이 다른 개발 요소에 위험요소로 전이되는 것을 예방하고 이를 예측하여 품질관리를 하려는 노력이 대두되고 있다. 따라서 본 논문에서는 불확실성을 관리하기 위하여 불확실성 판단 기준을 제시하고 발생된 불확실성을 없애기 위한 불확실성 처리 프로세스를 제안한다. 또한 불확실성의 개선 정도를 정량적으로 분석하기 위하여 불확실성의 정량적인 판별을 수행하고자 한다.

† 종신회원 : 안동대학교 컴퓨터공학과 부교수
Manuscript Received : December 2, 2014
First Revision : January 6, 2015
Accepted : January 7, 2015
* Corresponding Author : Eun-Ser Lee(eslee@anu.ac.kr)

2. 관련 연구

2.1 소프트웨어 신뢰성

개발되는 소프트웨에서 테스팅하고 있는 소프트웨어에 얼마나 많은 신뢰를 가질 수 있는지를 알기 위해서 결함추정을 이용할 수 있다. 보통 비율로서 표현되는 신뢰는 소프트웨어가 결함을 지니지 않을 가능성을 알려준다. 그러므로 만약 프로그램을 95% 신뢰수준에는 결함이 없다고 말한다면 그것은 소프트웨어가 결함을 가지지 않을 가능성이 0.95임을 의미한다. 프로그램에 S개의 결함을 투입하면서 실제 결함은 단지 N개로 알고 있다고 가정해 보자. 우리는 모든 S개의 투입 결함들을 찾을 때까지 프로그램을 테스트한다. 만약 이전대로 n이 테스팅 중에 발견된 실제 결함의 개수라면 신뢰수준은 다음과 같이 계산될 수 있다[13].

$$\begin{Bmatrix} = 1, & \text{if } n > N \\ = S/(S-N+1), & \text{if } n \le N \end{Bmatrix} \quad (1)$$

이와 같은 방법으로 결함을 관리하여 신뢰수준을 높일 수 있다.

추정 기법은 우리에게 프로그램에 대한 신뢰를 줄뿐만 아니라 부수적인 이점도 제공한다. 많은 프로그래머들은 각 결함이 마지막인 것으로 마무리 짓고 싶어 한다. 만약 남아있는 결함의 개수를 추정하거나 신뢰 요구사항을 만족시키기 위해서 얼마나 많은 결함들을 찾아야 하는지 알고 있다면, 좀 더 많은 결함에 대한 테스팅을 유지하는 것이 고무적일 것이다[13].

2.2 품질

품질에 대한 이해는 5가지 관점으로 기술할 수 있다.

- 품질은 우리가 인식할 수는 있으나 정의하지 않은 어떤 것이라는 관념적 관점(transcendental view)
- 품질은 목적에 대한 적합성이라는 사용자 관점 (user view)
- 품질은 명세서에 대한 순응성이라는 제조 관점 (manufacturing view)
- 품질은 고유의 프로덕트 특성에 얽매인다는 프로덕트 관점(product view)
- 품질은 고객이 프로덕트를 위해 지불하는 지불액에 의존한다는 가치기반 관점(value-based view)

관념적 관점은 플라톤의 이상(ideal)에 대한 기술이나 아리스토텔레스의 형식(form)의 개념과 매우 유사하다. 바꿔 말하면 모든 실제 목록은 이상적인 목록의 근사치인 것과 같이 우리가 노력하는 것에 대한 이상으로 소프트웨어의 품질을 생각할 수 있다. 그러나 결코 그것을 완전하게 구현할 수 없을 수도 있다. 관념적 관점은 더욱 구체적인 사용자 관점에 비하면 희미한 것이다. 전체 프로덕트 품질을 이해하기 위해 결함 밀도나 신뢰성과 같은 프로덕트 특성들을 측정할 때에는 사용자 관점을 취한다[12].

제조 관점은 프로덕트를 생산하는 동안과 인도한 후에 품질에 대해 조사한다. 특히 인도된 프로덕트의 결함을 고치기 위한 재작업에 드는 비용을 없애기 위해 초기에 올바르게 구축되었는지를 조사한다. 따라서 제조 관점은 양질의 프로세스에 순응하는 프로세스 관점이다. 그러나 프로세스에 대한 순응성이 프로덕트에서 보다 적은 결함과 오작동을 야기한다는 증거는 없다. 프로세스에 의해 사실상 고품질의 프로덕트에 이를 수도 있지만 평범한 프로덕트의 생산을 제도화하는 것이 가능할 수도 있다[12].

사용자 관점과 제조 관점은 외부에서 프로덕트를 조사하지만 프로덕트 관점은 내부를 자세히 살펴보고 프로덕트 고유의 특성을 평가한다. 이러한 관점은 소프트웨어 측정 전문가들이 종종 주장하는 관점으로 좋은 내부 품질 척도는 신뢰성이나 유지보수성과 같은 양질의 외부 품질을 이끌어낼 것이라고 추측한다. 그러나 이러한 추측에 대한 검증과 실제 프로덕트의 사용에 영향을 미치는 품질을 결정하기 위해서는 더 많은 연구가 필요하다. 그러므로 프로덕트 관점과 사용자 관점을 연결하는 모델을 개발해야만 할 것이다[12][13].

개발자와 경영진은 다른 관점으로 품질을 평가한다. 따라서 품질에 대한 합의가 이루어지지 않는다면 제품에 대한 품질의 완성도가 떨어지게 된다.

2.3 위험분석

위험이 소프트웨어 프로젝트에만 한정된 것은 아니다. 소프트웨어가 성공적으로 개발되고 고객에게 인도된 후에도 위험은 일어날 수 있다. 이러한 위험은 보통 현장에서의 소프트웨어 실패와 관련이 있다[7].

비록 잘 공학화 된 시스템이 실패할 확률이 낮다고 해도, 컴퓨터-기저 제어 또는 감시 시스템에서 검출되지 않은 결점은 막대한 경제적 손실을 초래하거나 사람이 부상하거나 죽음을 초래할 만큼 더욱 나쁘고 심각할 수 있다. 그러나 컴퓨터-기저 제어와 감시 시스템 비용 상의 이점과 기능상의 이점은 보통 이러한 위험보다 중요하다. 오늘날 컴퓨터 소프트웨어와 하드웨어가 안전을 증시하는 시스템을 제어하기 위해 사용되고 있다.

소프트웨어 안정성과 위험분석은 소프트웨어에 부정적으로 영향을 미치는, 그리고 전체 시스템을 작동하지 않게 만드는 잠재적인 위험의 식별과 평가에 초점을 맞춘 소프트웨어 품질보증활동이다. 만약 소프트웨어 공학 프로세스의 초기에 치명적 위험을 식별할 수 있다면, 소프트웨어 설계 특징들을 명시할 수 있고, 잠재적인 치명적 위험을 제거하거나 조정할 수 있다[8].

3. 본론 및 사례연구

소프트웨어 개발 및 프로젝트 수행에서 성패 여부는 많은 요인으로 결정되지만 원인이 해결되지 않는 불확실성에 대해서는 치명적인 결과를 초래할 수 있다. 예를 들어 요구사

항 정의서의 결함이 설계 단계에서 결함으로 발생하는 경우 요구사항단계에서 결함 요인을 인식하지 못하고 다음 단계인 설계 단계에서도 결함의 원인을 판별하지 못하는 경우가 많이 발생된다. 이와 같은 문제점들을 해결하기 위하여 본 장에서는 과제 수행 시 발생하는 불확실성을 줄이고 이를 판별하기 위하여 불확실성을 관리하고자 한다.

불확실성은 결함과는 달리 가시화되지 않을 수 있는 것으로서 어떤 특성과 내용을 내포한가에 대한 판별이 어려울 수 있다. 또한 불확실성은 과제 초기에 기능을 명세하면서 발생되는 경우가 대부분이다. 이와 같은 불확실성은 기능의 정확한 파악을 어렵게 만들고, 전체 시스템에 그 영향을 미칠 수 있게 된다. 즉, 정성적인 부분을 많이 내포하고 있어서 정량적으로 표현하기 어려운 경우이다. 정성적인 부분을 많이 내포하기 때문에 기능이 어느 정도 완전하게 완성이 되는지와 정확하게 기능을 수행하는지를 판별하기가 어렵게 된다.

결함은 현재 상태에서는 문제로 나타나지는 않았지만 향후 비용, 일정, 품질에 영향을 주게 된다. 불확실성도 향후 비용, 일정, 품질에 영향을 줄 수 있지만 현재 기능에 대하여 내용을 파악할 수 없어서 특성을 결정할 수 없거나 기능을 판별할 기준이 모호한 경우 등이 해당된다. 결과적으로 불확실성은 과제를 수행하는 기간 동안 해결이 되지 않을 수도 있다.

본 논문에서는 불확실성을 해결하여 관리하기 위한 기준 및 처리를 위한 프로세스, 불확실성을 정량적으로 판별하는 방법에 대하여 제시하고자 한다.

3.1 불확실성의 판단 기준

불확실성은 많은 모호성 내포하고 있으며, 모호성은 다양한 형태를 가지고 있다. 이와 같은 모호성과 다양한 형태는 기능이 완전하게 구현되는 것을 방해하는 요인이 된다. 따라서 모호성을 제거하는 것이 불확실성을 경감시키는 방법이 된다.

불확실성을 줄이기 위한 노력은 다양한 형태로 표출이 된다. 그러나 불확실성이 발생한 현상을 중심으로 문제를 해결하는 것도 하나의 방법이지만 근본적인 해결책이 될 수는 없다. 따라서 불확실성의 원인을 찾아서 이를 제거하기 위하여 불확실성의 다양한 형태를 정의한다. 이와 같은 정의는 불확실성 여부를 판별하기 위해서는 기준 설정이 선행되어야 한다. 불확실성을 관리하기 위하여 범주를 정의하고 판단기준을 정의한다. 또한 처리 프로세스를 구성하고 이를 기반으로 정량적인 판별을 하여 불확실성을 관리하고자 한다.

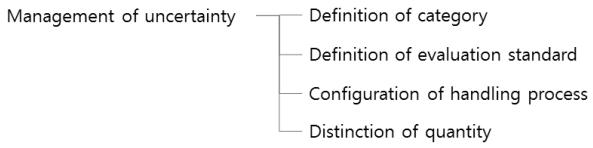

Fig. 1. Configuration factors for management of uncertainly

불확실성을 판단하기 위하여 기준을 제시하였다. 판단 기준은 기능을 명세하고 구현 및 측정을 하는 범주로 제시를 하였다.

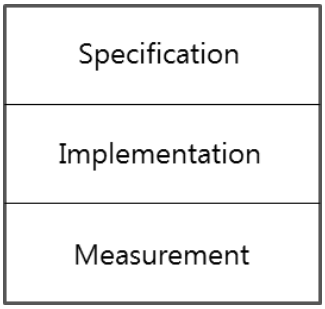

Fig. 2. Category of uncertainty

명세는 초기의 요구사항을 추출하여 기능화 하는 과정에서 발생하는 불확실성을 의미한다. 따라서 요구사항 추출, 요구사항 정의, 모델링 등이 해당되게 된다.

구현은 설계를 기반으로 개념적인 내용을 구체화 하는 과정을 의미한다.

측정은 완성된 구현물이 명세와 일치하는지를 확인하고 제 기능을 수행하는가를 측정하는 것이다.

Fig. 2에서의 범주를 기반으로 하여 불확실성을 판단하는 기준을 제시하였다. 불확실성 판단 기준은 다음과 같다.

1) 불확실성 판단 기준
① 개념 및 원리의 불일치
② 형태나 구조를 정형화 할 수 없는 경우
③ 기본 특성을 명세할 수 없는 경우
④ 명세된 특성을 정량적으로 표현할 수 없는 경우
⑤ 결과의 예측이 불가능한 경우
⑥ 입력 값의 측정이 불가능한 경우
⑦ 출력 값의 측정이 불가능한 경우
⑧ 데이터가 일관성 있는 값을 산출하지 못하는 경우
⑨ 측정 오차를 벗어나는 경우
⑩ 문제의 원인을 파악할 수 없는 경우

제시한 불확실성 판단 기준은 명세, 구현, 측정의 범주로 구분하여 불확실성의 원인을 확인하려고 한다. 구분은 다음과 같다.

- 명세 : ①, ②, ③, ⑩
- 구현 : ②, ④, ⑩
- 측정 : ⑤, ⑥, ⑦, ⑧, ⑨, ⑩

구분된 불확실성 판단 기준에서 “② 형태나 구조를 정형화 할 수 없는 경우”는 명세와 구현의 범주에 속하게 된다. 또한 “⑩ 문제의 원인을 파악 할 수 없는 경우”는 명세, 구현 및 측정의 범주에 모두 속하게 된다. 따라서 모든 기능과 이슈를 제시한 범주와 불확실성 판단 기준에 의하여 구분을 한다.

3.2 불확실성 처리 프로세스

불확실성의 발생 빈도를 줄이기 위해서는 불확실성이 발생하기 위한 범주와 판단기준을 제시하였다. 발생된 불확실성은 프로세스를 통하여 관리를 하고자 한다. 불확실성을 처리하기 위해서는 다음과 같은 프로세스를 구성하여 관리하고자 한다. 불확실성 처리 프로세스는 분석, 예방, 문제 해결, 대안 프로세스로 구성하였다. 각 프로세스의 정의는 다음과 같다.

- 분석 프로세스 : 불확실성을 식별하기 위한 개념 정의, 명세, 연관성 분석 등의 기본 특성을 결정하기 위한 프로세스
- 대안 프로세스 : 불확실성이 명확한 특성과 형태를 갖지 못하는 경우에 차선책을 적용하여 진행하기 위한 프로세스
- 문제 해결 프로세스 : 발생된 불확실성이 명확한 특성을 갖게 위한 프로세스
- 예방 프로세스 : 사전에 불확실성을 예방하기 위한 프로세스

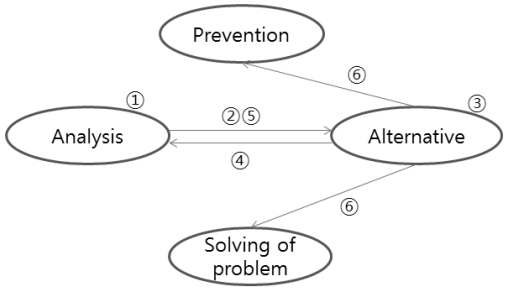

Fig. 3. Process diagram for management of uncertainly

불확실성 관리를 위해서 분석, 대안, 문제 해결 및 예방 프로세스를 구성하였다. Fig. 2에서는 불확실성 관리를 위한 프로세스 구성도를 나타내고 있다. Fig. 3에서 분석 프로세스는 불확실성을 식별하고 분석하게 된다(①). 이와 같은 자료들을 대안 프로세스에 제공하여(②) 불확실성을 해결하기 위한 차선책을 만들 수 있게 해준다. 대안 프로세스에서는 대안을 만들어서(③) 분석 프로세스에 보내어(④) 식별된 불확실성에 효과적으로 대안이 되는가를 판별하게 된다. 대안으로서 효용성이 있다면 이를 대안 프로세스에 최종적으로 전달한다(⑤). 대안 프로세스는 예방 프로세스에 전달하여 전체 시스템에 적용하여 같은 불확실성이 다시 발생하지 않도록 한다(⑥). 또한 대안프로세스는 대안을 발생된 불확실성을 제거하기 위하여 문제해결 프로세스에 대안을 전달하여 해당 불확실성을 제거하게 된다(⑥). 이와 같은 프로세스를 유기적으로 연결하여 불확실성을 프로세스 관점에서 관리하고자 한다.

3.3 불확실성의 정량적인 판별

불확실성을 식별하여 원인을 분석하고 대안을 세워서 해결하기 위해서는 불확실성이 맞는지를 판별하는 것이 중요한 요소가 된다. 정상적인 기능을 수행하는 요소를 잘못 판단하여 불확실성 요소로 판단한다면 전체 시스템에서 큰 위험 요소가 될 것이다. 이와 같은 위험요소는 일정, 프로젝트 비용, 품질 측면에서 위험 요소가 되어서 시스템의 신뢰성을 떨어뜨리게 된다. 따라서 불확실성 요소를 정확하게 판별하기 위하여 정량적인 판별의 방법으로 접근하고자 한다. 불확실성의 정량적인 판별을 위하여 잔여 불확실성의 비율, 불확실성 판단(표준편차에 의한 측정 오차에 의한 판단)으로 산출하도록 하였다.

잔여 불확실성 비율은 완성도와 불확실성 개선 비율에 의하여 불확실성을 판별하고 있다. 잔여 불확실성 비율은 다음과 같다.

1) 잔여 불확실성 비율

$$\text{불확실성} = \sum_{i=1}^{j} U(f_i) \text{ X 완성도 (U는 상호 배타적인 요소 (job 또는 task))}$$

$$\text{완성도} = \frac{\text{일관성있게 수행 완료되는 기능}}{\text{전체 세부 기능}} \times 100 \quad (2)$$

$$\text{불확실성의 개선 비율} = \frac{\text{현재 상태의 불확실성 개수}}{\text{이전 상태의 불확실성 개수} + \text{현재 상태의 불확실성 개수}} \times 100$$

불확실성은 상호 배타적인 요소의 완성도로 판별한다. 완성도는 전체 세부 기능 중에서 일관성 있게 기능 수행을 완료하는 비율을 의미한다. 그리고 이를 기반으로 전체 불확실성의 개수 중에서 현재 상태의 불확실성이 차지하는 비율을 산출하여 불확실성의 개선 비율을 얻을 수 있다.

4. 사례 연구

이론을 적용하여 확인하기 위하여 과제를 선정하여 확인하였다.

개발 과제는 노년층을 위한 스마트폰 앱을 개발하는 것으로서 기능은 돋보기, 사진 보정, 음성 출력의 기능을 대상으로 하고 있다. 개발 과제의 유스케이스는 다음과 같다.

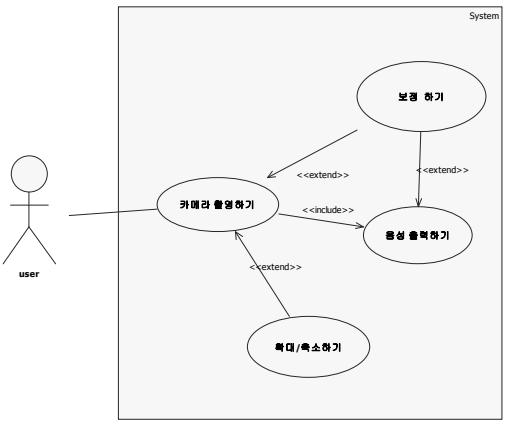

Fig. 4. Use case

1) 시나리오
- 카메라 촬영하기 : 사용자는 application을 실행시켜 카메라를 촬영한다.
- 음성 출력하기 : 사용자는 보정된 이미지를 음성으로 출력한다.
- 확대/축소하기 : 사용자는 촬영된 이미지를 확대/축소 한다.
- 보정하기 : 사용자는 카메라 촬영할 때 보정하기 기능으로 올바른 이미지를 출력한다.

사용자는 보정하기 기능으로 촬영된 이미지를 음성으로 출력한다.

개발 과제의 요구사항 명세서에 의하여 각각의 요구사항들을 3.1절에서 제시한 불확실성 기준에 의하여 구분하였다. 구분한 결과는 다음과 같다.

2) 이야기 목록제공

③ 기본 특성을 명세할 수 없는 경우
- 원인 : 목록화를 어떤 버튼을 사용할 것인지 사용자의 편의를 위해서라면 어떤 버튼의 모양, 크기, 위치 등을 할 것인지.
- 대처 방안 : 보편화된 UI에 따름(버튼 형태로 가운데 정렬하여 사용자가 선택할 수 있게 함).

3) 사진 보정 기능(잘라내기)

② 형태나 구조를 정형화 할 수 없는 경우
- 원인 : 어떤 크기, 모양으로 잘라낼 것인가?
- 대처 방안 : 사진을 잘라내는 API를 기초로 해서 클래스 다이어그램을 수정하여 개발

⑩ 문제의 원인을 파악할 수 없는 경우
- 원인 : 코딩 오류 및 옮기기 기능과 충돌
- 대처 방안 : 옮기기 기능과 화소를 동일하게 설정

4) 사진 보정 기능(옮기기)

① 개념 및 원리의 불일치
- 원인 : 얼굴만 잘라놓은 사진이 아니라 전신 사진일 경우
- 대처 방안 : 붙여 넣을 공간의 이미지 크기 값을 지정하여 해당크기만 붙여넣기가 가능하게 설정(ex 3*4)

② 형태나 구조를 정형화 할 수 없는 경우
- 원인 : 사진을 잘라내는 도구가 정형화되지 않음.
- 대처 방안 : 사진을 붙여 넣는 API를 기초로 하여 클래스 다이어그램을 수정하여 코딩

⑩ 문제의 원인을 파악할 수 없는 경우
- 원인 : 코딩 오류 및 잘라내기와 충돌
- 대처 방안 : 잘라내기 기능과 화소를 동일하게 설정.

5) 사진 보정 기능(가장자리 다듬기)

① 개념 및 원리의 불일치
- 원인 : 동화에 사진을 붙일시 일그러짐 현상(톤, 자세, 위치, 명암)
- 대처 방안 : 사용자가 동화에 맞는 포즈의 사진으로 자르고 밝기 색깔의 명암 설정이 되게 설정

② 형태나 구조를 정형화 할 수 없는 경우
- 원인 : 테두리, 공백 정형화 된 것이 없음.
- 대처 방안 : 편집 기능의 API를 기초로 하여 클래스 다이어그램을 수정하여 코딩

③ 기본 특성을 명세할 수 없는 경우
- 원인 : 어떤 편집 기능 들을 사용할지?
- 대처 방안 : 밝기, 색깔의 진하고 연함을 편집하는 기능으로 코딩

⑩ 문제의 원인을 파악할 수 없는 경우
- 원인 : 이미지의 경우 반드시 고화질, 중화질, 저화질 중 선택하여야함.
- 대처 방안 : 해상도를 일관성 있게 설정하여 최적화.

6) 사용법

① 개념 및 원리의 불일치
- 원인 : 스마트폰을 처음 이용하는 분들은 설명서가 있어도 사용의 어려움
- 대처 방안 : 유치원생들도 따라 할 수 있을 정도의 간편한 UI와 쉬운 설명으로 구성

② 형태나 구조를 정형화 할 수 없는 경우
- 원인 : 스마트폰에 익숙하지 않은데 어떤 UI로 설명서를 만들지 선택.
- 대처방안 : 기존의 설명서들을 참고하고, API를 기초로 하여 코딩.

발생된 불확실성을 분류한 결과를 토대로 3.2절의 프로세스를 적용하여 불확실성을 제거하였다. 제거한 결과는 기능별로 다음과 같으며, '기능 - 원인 - 대처 방안'으로 나열하였다.
- 이야기 목록제공 - 어떤 버튼의 모양, 크기, 위치 등을 할 것인지 미설정 - 보편화된 UI를 따라 디자인(가운데, 버튼 정렬)
- 사진 보정 기능 (잘라내기) - 어떤 크기, 모양으로 잘라낼 것인가? - 사진을 잘라내는 API를 기초로 해서 클래스 다이어그램을 수정하여 개발
- 사진 보정 기능 (잘라내기) - 문제의 원인을 파악할 수 없는 경우 - 옮기기 기능과 화소를 동일하게 설정
- 사진 보정 기능(옮기기) - 얼굴만 잘라놓은 사진이 아니라 전신 사진일 경우 - 사진의 크기를 고정
- 사진 보정 기능(옮기기) - 사진을 잘라내는 도구가 정형화되지 않음 - 사진을 붙여넣는 API를 기초로 하여 클래스 다이어그램을 수정하여 코딩
- 사진 보정 기능(옮기기) - 코딩 오류 및 잘라내기와 충돌 - 잘라내기 기능과 화소를 동일하게 설정
- 사진 보정 기능(다듬기) - 동화에 사진을 붙일 시 일그러짐 현상 - 사용자가 동화에 맞는 포즈의 사진으로 설정 유도(밝기 색깔, 명암 설정, 좌우반전 효과)
- 사진 보정 기능(다듬기) - 테두리, 공백 정형화 된 것이 없음 - 편집 기능의 API를 기초로 하여 클래스 다이어그램을 수정하여 코딩

- 사진 보정 기능(다듬기) - 어떤 편집 기능 들을 사용할지? - 기능으로 밝기, 색깔, 좌우대칭을 편집하는 기능으로 설정
- 사진 보정 기능(다듬기) - 이미지의 경우 반드시 고화질, 중화질, 저화질 - 해상도를 일관성 있게 설정하여 최적화
- 사용법 - 스마트폰을 처음 이용하는 분들은 설명서가 있어도 사용의 어려움 - 처음 사용하는 유저도 따라할 수 있을 정도의 간편한 UI와 쉬운 설명으로 구성
- 사용법 - 스마트폰에 익숙하지 않은데 어떤 UI로 설명서를 만들지 선택 - 기존의 설명서들을 참고하고, API를 기초로 하여 코딩

불확실성 기준과 불확실성 처리 프로세스에 의하여 대처 방안을 만들었다. 따라서 기존의 불확실성을 제거하기 위하여 노력하였다. 이와 같은 결과를 기반으로 하여 불확실성 개선 비율을 분석하였다.

초기의 완성도는 전체 세부 기능 대비 일관성 있게 실행되는 기능이 약 33%였다(전체 세부 기능 6개, 실행되는 기능 2개). 불확실성 기준과 처리 프로세스의 대안에 의하여 발생된 문제를 처리한 결과 불확실성 개선 비율은 전체 불확실성 24개 중에서 실제로 6개가 발생되었다. 따라서 불확실성 개선 비율은 약 21.4%가 되었다. 결론적으로 발생된 불확실성 요소들은 불확실성 판단기준과 불확실성 처리 프로세스 및 정량적인 관리에 의하여 감소되는 것을 확인하였다.

5. 결론 및 향후 연구 방향

본 논문에서는 불확실성이 위험요소로 전이되는 것을 예방하여 전체 소프트웨어의 신뢰성을 높이고자 불확실성 판단기준, 불확실성 처리 프로세스 및 불확실성의 정량적인 판별을 제시하였다. 따라서 불확실성이 위험요소로 전이되는 부분을 제거하여 전체 소프트웨어의 신뢰성을 유지하고 향상할 수 있었다.

향후 연구로는 제시된 기준을 개발 과제의 영역별로 세분화하여 적용할 수 있도록 제시할 필요가 있다. 또한 이를 지원하기 위하여 자동화 도구를 개발하여 불확실성 관리를 효과적 및 효율적으로 할 수 있도록 개발할 계획에 있다.

References

[1] Yoon chung, Successful Software development methodogy, Life power press, 1999.08.

[2] Huh won sil, System analysis and design, Hanbit media, 2006.05.

[3] Software process improvement forum, KASPA SPI-7, 2002.

[4] McCall, P. K. richards and G. F. Walters, “Factors in software quality”, Vol.1, 2 and 3, Springfield VA, AD/A-049-014/015/055, 1997.

[5] Wohlin, Runeson, “Defect content estimations from review data”, Proceedings international conference on software engineering ICSE, pp.400-409, 1998.

[6] Gaffney, John, “Some models for software defect analysis”, Lockheed martin, 1996.

[7] L. Hatton, “Is modularization always good idea”, Information and software technology, Vol.38, pp.719-721, 1996.

[8] B. Compton and C. Withrow, “Prediction and control of ada software defects”, J. systems and software, Vol.12, pp.199-207, 1990.

[9] Roger, S. Pressman, “Software engineering”, Mcgraw-hill international edition, 1997.

[10] Choi eun man, Software engineering, Jungik publishing co, 2011.

[11] Charles conrick IV, Scott hanson, Vertical option spreads : a study of the 1.8 standard deviation inflection point, Hoboken, N.J. : Wiley, 2013.

[12] Ian Sommerville, Software engineering, Addison Wiley, 2008.

[13] Shari Lawrence Pfleeger and Joanne M. Atlee, Software engineering theorey and practice, Pearson, 2013.

이 은 서

e-mail : eslee@anu.ac.kr

2001년~현 재 ISO/IEC 15504 국제선임심사원

2004년 중앙대학교 컴퓨터공학과(박사)

2004년~현 재 임베디드산업협회 전문위원

2004년~현 재 한국정보통신기술협회 위원

2012년~현 재 안동대학교 컴퓨터공학과 부교수

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