KIPS Tr. Software and Data Eng.
Vol.3, No.10 pp.389~396 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2014.3.10.389

Design of Checklist for Improvement of Reliability of Uncertainty and Variability Environment

Eun-Ser Lee†

ABSTRACT

There are many defects when we're working on a project. Especially, if we don't have experience in similar field, defects have a strong influence on the system entirely. Therefore, management of risk such like defects is important the factor for success of project. Finally, efficient management of risk can guarantee handling problems such like fault tolerancy, unexpected quality and delay of schedule. In this paper, we propose checkpoint for improvement of reliability and are willing to improve the reliability of an entire system using it.

Keywords : Checkpoint, Risk Management, Improvement of Reliability

불확실성 및 가변성 환경의 신뢰성 향상을 위한 체크리스트 설계

이 은 서†

요 약

프로젝트 수행은 많은 결함을 내포하고 있다. 특히 유사한 분야의 프로젝트 수행 경험이 없다면 결함은 전체 시스템에 많은 영향을 주게 된다. 따라서 이와 같은 결함은 프로젝트의 성공 여부를 판단할 수 있는 중요한 요소가 된다. 결국 효율적인 위험관리는 문제에 쉽게 대처할 수 있게 해주며, 예상하지 못한 품질, 예산초과 및 일정 지연이 되지 않도록 해준다. 본 연구에서는 소프트웨어 개발 시, 프로세스의 신뢰성 향상을 위하여 체크포인트를 제시하고 이를 활용하여 전체 시스템의 신뢰성을 향상하고자 한다.

키워드 : 체크포인트, 위험관리, 신뢰성 향상

1. 연구배경

소프트웨어 사용의 증가에 따라서 이를 사용하고 제어하기 위하여 비용이 증가되고 있다[1]. 많은 소프트웨어가 서로 다른 목적에 의하여 만들어지고 있으며, 이와 같은 상황에서 소프트웨어들의 연동이 필수 요소가 되고 있다. 각각의 목적에 의해서 만들어지는 소프트웨어가 연동이 되어서 사용된다는 것은 개별 소프트웨어가 결함을 내포하지 않는다는 전제를 필요로 한다. 결함을 내포하고 있으면 다른 소프트웨어에 잘못된 결과나 제어명령을 전달하여 기대하지 않은 결과를 얻게 될 수 있다. 따라서 소프트웨어 신뢰성을 제공하기 위하여 많은 노력이 이루어지고 있다[2]. 소프트웨어 신뢰성에 영향을 주는 많은 요인 중에서 불확실성과 가변성은 소프트웨어를 분석하고 설계하는 과정에서 필수적으로 확인을 하여 제거를 해야 하는 요소이다.

불확실성과 가변성을 제거하기 위하여 소프트웨어 검토를 수행하며, 이와 같은 활동은 분석, 설계 및 코딩이라 불러온 소프트웨어 엔지니어링 활동을 정화시킨다. 따라서 검토는 소프트웨어 개발 동안 다양한 지점에 적용되며 오류들과 결함들을 발견하는 일을 한다[3].

본 논문에서는 불확실성과 가변성이 발생할 수 있는 환경에서 신뢰성을 높이기 위하여 체크리스트를 제안하고자 한다. 제안된 체크리스트는 요구사항을 정의하는 단계에 대하여 정의를 한다. 요구사항 정의 단계의 세부과정은 3단계로 구성을 하였다. 사용자 요구사항 정의, 환경 요구사항 정의, 요구사항 중에서 불확실성 및 가변성 분류 단계이다. 제안되는 세 단계에 의하여 체크리스트를 작성하고 이를 기반으로 정량적인 측정을 수행하게 된다. 측정된 결과 값에 의하여

※ 본 논문은 2013학년도 안동대학교 국제학술교류보조금에 의하여 연구되었음.
† 종신회원 : 안동대학교 컴퓨터공학과 부교수
Manuscript Received : August 26, 2014
Accepted : September 4, 2014
* Corresponding Author : Eun-Ser Lee(eslee@anu.ac.kr)

불확실성과 가변성의 결함 부분이 식별된다. 식별된 결과를 기반으로 하여 원인과 대안을 작성하여 프로젝트를 구성하는 프로세스에 적용하게 된다. 따라서 불확실성과 가변성을 제거할 수 있게 되고 전체 시스템의 신뢰성을 향상할 수 있게 된다.

2. 기본 개념

2.1 결함

품질은 많은 의미가 담긴 다차원 개념이다. 이로 인하여 파생되는 두 가지 중요한 결과가 있는데 하나는 소프트웨어 품질은 하나의 숫자로 축약할 수 없다는 것이다. 둘째는 품질 개념은 프로젝트마다 다르다는 것이다. 아주 민감한 프로젝트에서 신뢰성은 가장 중요할 수 있지만 사용 용이성은 그리 중요하지 않다. 소프트웨어 개발 프로젝트마다 개발하기 전에 품질 목표를 설정해야 하고 개발 프로세스의 목표도 품질 목표를 만족시켜야 한다[4][5].

많은 품질 요인이 있지만 일반적으로 신뢰성이 소프트웨어 품질의 기준을 대표한다. 소프트웨어가 신뢰성이 없다는 것은 소프트웨어에 결함이 존재하기 때문이다. 품질을 측정하는 한 가지 방법은 출시된 소프트웨어의 단위 크기당 결함의 수이다. 이러한 측면에서 볼 때 품질 목표는 할 수 있는 한 KLOC당 결함의 수를 줄이는 것으로 귀결된다. 소프트웨어 공학의 가장 좋은 기법으로 KLOC당 1개 미만으로 줄일 수 있다[6].

품질에 대한 정의를 사용하려면 결함을 명확히 정의하여야 한다. 무엇을 결함으로 간주할 것인지는 프로젝트나 프로젝트를 진행하는 기관이 사용하는 표준에 따라 다르다.

2.2 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 공학화된 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 시스템의 비용상의 이점과 기능상의 이점은 보통 이러한 위험보다 중요하다. 오늘날 컴퓨터 소프트웨어와 하드웨어가 안전을 중시하는 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].

2.3 불확실성 요소

소프트웨어 품질 모델은 각 모듈을 신뢰성 있는 통계방법으로 예측할 수 있는 것을 의미한다[9]. 이와 같은 모델은 결함을 찾아서 해결하기 위한 것으로 초기의 소프트웨어 생명주기에 주로 적용을 했다.

성공적인 소프트웨어를 개발하기 위하여 유용한 방법론들, 기술들 그리고 도구 등이 사용된다. 따라서 소프트웨어 개발자들은 소프트웨어 구성요소들 간의 관계와 규칙 제약사항을 이해하고 있어야 한다[10][11]. 만약 개발자들이 각 요소의 내용을 이해하고 있지 못하다면 개발 과정에서 불확실성을 내포하게 된다. 따라서 불확실성을 제거하기 위하여 중요한 요소가 존재하게 된다. 그 요소는 기술과 방법론에 대하여 명확히 이해를 하는 것이다. 이를 위해서 소프트웨어 프로세스에 대한 지식과 이해가 필요하게 된다.

2.4 가변성 요소

소프트웨어 프로젝트의 가변성은 외부와 내부요인으로 나눌 수 있다. 외부요인은 문제, 사용자, 정보, 개발조직, 고객의 제약사항이 있다. 내부요인으로는 프로세스 방법과 도구, 개인의 능력, 산출물이 있다. 이와 같은 외부와 내부의 요인이 소프트웨어 프로젝트를 구성하는 중요한 요인이며, 많은 가변성을 발생시킬 수 있는 부분이 된다.

가변성의 문제는 정의, 사용자 요구의 변경, 영역의 문제, 영역의 복잡도, 작업의 복잡도, 응용분야의 형태로 발생하게 된다. 따라서 가변성의 제거 문제는 앞에서 제시된 문제점을 해결하는 것을 의미한다[12].

본 논문에서는 불확실성과 가변성을 제거하기 위하여 체크리스트를 제시하고 이를 해결하여 전체 시스템의 신뢰성을 향상시키고자 한다.

3. 본론 및 사례연구

본 장에서는 개발 과정에서 발생하는 불확실성과 가변성을 줄이기 위하여 체크리스트를 제시하고 있다. 불확실성과 가변성을 찾아서 최종 상태를 정의해서 다른 항목들에 전이

되는 영향 정도를 정형화하여 프로세스 관점으로 관리하고자 한다. 따라서 이와 같은 활동을 위하여 요구사항을 정의하고 불확실성 및 가변성을 확인, 정의하는 과정이 체크리스트의 주요한 이정표가 된다. 따라서 불확실성과 가변성의 특성이 있는 프로젝트의 경우에 적용할 수 있는 체크리스트 설계를 위하여 프로세스의 기본적인 설정 기준을 마련하고자 한다. 또한 불확실성과 가변성을 위한 체크리스트를 설계하여 프로젝트 수행의 완성도를 높이고자 한다.

3.1 프로세스의 기본적인 기준 설정

불확실성 및 가변성 환경에서 사용할 체크리스트를 설계하기 위해서는 체크리스트를 구성하는 프로세스들의 기본적인 기준을 설정하는 것이 선행되어야 한다. 프로세스의 기본적인 기준을 설정하는 것은 전체 체크리스트의 특성을 결정할 수 있고, 프로세스의 일관적인 성능을 얻을 수 있게 해준다.

체크리스트를 구성하는 프로세스들은 다음과 같이 세 가지 특성을 기본적인 기준으로 제시하였다.

Table 1. Criteria of processes

QSUG
CMA
CCA

세 가지 특성은 정량화 가능한 목표 설정(Quantitative Set Up Goal: QSUG)과 검증된 방법 사용 여부(Certification Method Availability: CMA), 검증된 기준 적용 여부(Certification Criteria Availability: CCA)로 구성되어 있다.

QSUG는 프로세스를 정의 시, 최종 결과물의 목표를 정량적으로 서술하여 제시할 수 있는가에 대한 내용이다. 목표 설정 시, 정량적인 것만 존재할 수는 없다. 그러나 이와 같이 정량적으로 서술 가능한 목표를 프로세스 기준 설정에 추가한 이유는 적용 프로젝트가 불확실성과 가변성을 위한 것이므로 정성적인 목표 설정에 의하여 목표가 도달되었는지에 대한 모호성을 줄이기 위함이다. 이와 같은 설정은 목표 도달 여부를 정량적으로 표현하여 진행 여부를 객관적으로 일관성 있는 판단을 제공할 수 있게 된다.

CMA는 프로세스를 구성하는 공정들의 수행 방법 선택 시, 방법론에서 발생할 수 있는 오류 및 결함을 줄이기 위하여 검증된 방법을 선택함으로써, 프로세스 구성의 완성도를 높일 수 있게 된다. 이와 같은 완성도는 QSUG의 정량적인 목표 설정을 가능하게 할 수 있다.

CCA는 CMA에서 적용한 방법이 검증된 기준으로 적용되었는지 판단할 수 있게 된다. 따라서 CMA는 검증된 방법을 사용했는가에 중점을 두지만, CCA는 적용된 방법을 검증된 기준에 의하여 판단할 수 있도록 적용을 하는가를 식별하는 작업이 된다.

3.2 불확실성 및 가변성 추출을 위한 요구사항 정의

불확실성 및 가변성이 있는 프로젝트 수행 시에 적용할 체크리스트는 기본적으로 필요로 하는 요구사항이 있다. 따라서 체크리스트를 설계하기 위하여 체크리스트의 주요한 이정표를 설정하고자 한다. 이정표는 요구사항 정의 단계, 불확실성 및 가변성 확인 단계, 불확실성 및 가변성 정의 단계로 분류하였다. 이와 같이 분류한 이유는 전체적으로 불확실성과 가변성을 추출하기 위하여 결함이 많이 포함되어 있는 요구사항부터 관리하기 위함이다. 모든 불확실성과 가변성의 원인은 요구사항에 많은 부분이 내포되어 있고, 이를 초기 단계에서 식별하지 못한다면 개발에 끝나는 시점에 가까울수록 불확실성과 가변성을 추출하여 관리하기는 어렵기 때문이다. 결론적으로 이정표는 요구사항을 기반으로 하여 불확실성과 가변성을 확인하고 정의하는 단계로 구성하였다.

세 단계로 구성된 이정표 중에서 본 절은 첫 번째 단계로서 요구사항을 정의하는 단계에 대하여 정의를 한다. 요구사항 정의 단계의 세부과정은 3단계로 구성된다. 사용자 요구사항 정의, 환경 요구사항 정의, 요구사항 중에서 불확실성 및 가변성 분류 단계이다.

사용자 요구사항 정의 단계는 사용자가 요구하는 기능을 정의하기 위한 단계이다. 따라서 구축될 시스템에서 필요한 기능을 추출하여 최종적으로 요구사항을 정의하는 단계이다. 주로 기능적인 요구사항이 대상이 되며, 사용자의 기능과 함께 시나리오를 추가하여 기능 추출의 완성도를 높여야 한다. 불확실성과 가변성을 식별하기 위하여 사용자 요구사항 단계에서는 기능 명세 시 다음과 같은 내용을 분석하도록 하여 불확실성과 가변성을 식별하고자 한다. 분석항목으로는 기능 변화 가능성, 기능 완성의 정량화 분석, 기능 명세의 가능 여부이다.

기능 변화 가능성은 요구사항에서 명세 될 기능이 향후 변화할 가능성이 있는지를 확인하는 것이다. 변화 가능성이

있다면 기능 자체가 변화하는 것인지 인터페이스만 변화하는 것인지에 대해서 확인을 해야 한다. 또한 현재 요구사항의 입력물이 변화가 일어날 수도 있으므로 입력물에 대한 확인도 필요하게 된다. 그리고 요구사항 기능을 수행한 결과물이 변화가 되는가와 결과물을 활용하는 다른 요구사항이 변화된 결과물을 요구하는지도 확인을 해야 한다.

기능 완성의 정량화 분석에서는 완성된 기능이 정량적으로 완성 정도를 표현할 수 있는지를 파악하는 것이다. 기능의 완성도를 정량적으로 표현할 수 있다는 것은 기능이 불확실성과 가변성을 모두 파악해야만 가능하기 때문에 완성도를 정량화하여 표현하는 것으로 불확실성과 가변성을 회피하였는지 알 수 있게 된다. 완성도를 정량적으로 파악하기 위해서는 기능을 구성하고 있는 세부 기능의 개수가 필요하다. 또한 세부 기능의 중요도도 필요하게 되어서 가중치를 부여하여 완성도를 정량적으로 표현할 수 있게 된다.

기능 명세의 가능성에서는 요구사항을 기반으로 기능을 추출하기 위하여 명세를 진행하게 되는데 기능화할 수 없다면 불확실성과 가변성을 가지고 있는 요구사항으로 분류할 수 있다. 따라서 이와 같은 요구사항은 다른 요구사항과의 연계성을 추적하여 다른 요구사항에 불확실성과 가변성이 전이되지 않도록 해야 한다.

Table 2. Checkpoint for definition of user requirement

Items	Checkpoint
Probability of function change	Change of core function Change of interface Change of input Change of output
Analysis of quantitative of function completion	Identify importance of detailed function Identify number of detailed function Quantitative of function completion
Probability of function specification	Name of function Attribute of function Operation of function Relationship of other functions

환경 요구사항 정의 단계는 구축하고자 하는 시스템의 구동 환경 및 필요로 하는 하드웨어를 정의하는 단계이다. 요구되는 하드웨어는 기능 요구사항을 완성하기 위하여 필요한 하드웨어이다. 따라서 환경 요구사항 정의 단계에서는 기능 요구사항을 기반으로 하여 필요한 환경을 정의하고자 한다. 가장 중요한 사항은 환경 요구사항의 변화 가능성과 상호 관계 변경 가능성이다. 이와 같은 사항은 불확실성과 가변성이 발생하는 원인이 되므로 환경 요구사항을 정의하여 변화 가능성과 변경 가능성을 확인해야 한다.

변화 가능성은 환경 요구사항에 필요한 시스템이 기능 및 다른 외부 요인에 의하여 바뀔 가능성에 대하여 확인을 하는 것이다. 따라서 시스템을 구성하고 있는 아키텍처의 변경 가능성을 확인한다. 그리고 시스템을 운영하는 방법이 새로운 것에 의하여 변경되는지, 부분 변경이 필요한지를 확인한다. 마지막으로 시스템에서 제어 명령이 빈번히 발생하여 다른 구성요소의 독립적인 동작을 방해하는지를 확인한다.

프로젝트를 수행하기 전에 체크포인터에 의하여 사전에 확인을 하여 대안과 결함이 많이 내포되어 있는 부분을 확인한다. 체크포인터를 프로젝트에 동일한 방법으로 적용하기에는 프로젝트의 특성을 고려할 수 없으므로 프로젝트를 수행하기 전에 프로젝트의 특성에 따라서 평가항목에 대한 비중 또는 중요도를 부여한다. 상위 체크포인터인 기능 변화 가능성, 기능 완성의 정량화 분석, 기능 명세의 가능성에 대하여 총합 100을 기준으로 비율을 할당하게 된다. 그 후 각 세부 체크포인터 항목인 기능 변화 가능성의 경우 핵심 기능의 변화, 인터페이스의 변화, 입력물의 변화, 출력물의 변화를 총합 100을 기준으로 비율을 할당한다. 이와 같은 방법으로 각 세부항목을 확인하여 상위항목에 대한 비율을 산정하고 전체적으로 사용자 요구사항 정의 단계를 정량적으로 산출하여 관리할 수 있게 된다.

상호 관계 변경 가능성은 기존에 구축된 시스템 사항이 부분적으로 변경이 필요한 경우이다. 따라서 외부 기능 및 요인과 연관성에 의한 부분과 자체적인 기능 변경이 요구되는 사항이 있는지를 판별할 필요가 있다. 확인될 부분은 다음과 같다. 시스템 구성요소들의 연관성에 의한 변경 가능성을 확인하여 시스템 구성요소가 변경되는지를 파악한다. 또한 시스템 구성요소 간의 방법론이 불일치하는지를 확인하여 방법론 교체에 의한 시스템 구성요소가 변경되는지를 파악한다. 마지막으로 시스템 구성요소 간의 인터페이스 불일치에 의한 시스템 구성요소가 변경되는지를 파악한다.

Table 3. Checkpoint for definition of environment requirement

Items	Checkpoint
Probability of transition	Identify of control command Probability of change of system architecture Probability of change of system operation methodology
Probability of change of relationship	Change of relationship of system components Disagreement of system components methodology Disagreement of system components interface

환경 요구사항 정의 단계에서 상위 체크포인터인 변화 가능성, 상호 관계 변화 가능성에 대하여 총합 100을 기준으로 비율을 할당하게 된다. 그 후 각 세부 체크포인터 항목을 총합 100을 기준으로 비율을 할당한다. 이와 같은 방법으로 각 세부항목을 확인하여 상위항목에 대한 비율을 산정하고 전체적으로 환경 요구사항 정의 단계를 정량적으로 산출하여 관리할 수 있게 된다.

요구사항 중에서 불확실성 및 가변성 분류 단계는 사용자 요구사항 정의 단계를 기반으로 하여 요구사항을 분류하는 단계이다. 분류를 위하여 기준이 되고 있는 불확실성과 가변성에 대하여 먼저 정의를 하고자 한다. 불확실성은 요구사항의 기능이 정확히 식별되고 있지 않은 상태를 의미한다. 또한 같은 기능을 반복하여 명세를 하는 과정에서 다른 기능이 명세 될 가능성이 있는 것을 의미한다. 즉 모호성을 갖고 있는 기능이 해당된다. 그리고 요구사항에서 사용되는 자료나 입력물들이 모호한 경우에도 불확실성을 갖게 된다. 이를 기반으로 결과물도 예측 불가능하게 되어서 불확실성을 갖게 된다. 가변성은 환경 및 시스템, 요구사항의 기능이 고정된 형태와 역할로 존재하는 것이 아니라 조건에 따라서 향후 변화될 특성을 갖는 경우를 의미한다. 환경의 변화, 시스템의 변경, 인적 자원의 변경, 기능의 변경, 기능들 간의 연관관계 변경 등이 있다.

Table 4. Checkpoint for uncertainly and variability in the requirements

Items	Checkpoint
Uncertainly	Ambiguity of function specification Ambiguity of data Unpredictability of output
Variability	Change of environment Change of system Change of personal resource Change of function Change of relationship among functions

요구사항 중에서 불확실성 및 가변성 분류 단계에서 상위 체크포인터인 불확실성, 가변성에 대하여 총합 100을 기준으로 비율을 할당하게 된다. 그 후 각 세부 체크포인터 항목을 총합 100을 기준으로 비율을 할당한다. 이와 같은 방법으로 각 세부항목을 확인하여 상위항목에 대한 비율을 산정하고 전체적으로 요구사항 중에서 불확실성 및 가변성 분류 단계를 정량적으로 산출하여 관리할 수 있게 된다.

3.3 체크포인트 항목의 불확실성 비율

체크포인트는 전체 3개의 큰 항목으로 구성을 하였다. 각각의 체크포인트 항목은 세부항목으로 구성을 하였다. 따라서 체크포인트 항목을 기준으로 하여 불확실성의 위험성이 얼마나 내포되어 있으며 비중이 높은 결과에 대하여 대안을 우선적으로 세울 수 있게 된다.

체크포인트의 연산은 다음과 같다.

1) 전체 3개의 체크포인트 항목에 프로젝트의 성격에 따라서 비중을 할당한다. 이때 전체 총합은 100을 기준으로 한다. 비중 할당은 각 항목의 중요도에 의하여 정성적으로 할당하게 된다.
2) 각각의 체크포인트 항목에 대한 세부항목에 대해서도 비중을 할당한다. 세부항목 비중의 총합은 상위항목의 비중을 초과할 수 없다. 비중 할당은 각 항목의 중요도에 의하여 정성적으로 할당하게 된다.
3) 각 항목에 실제 프로젝트를 기준으로 값을 입력한다. 각 항목은 1부터 10까지 입력을 한다. 1은 낮음이며 10으로 갈수록 높음을 의미한다.

Table 5. Result of user requirements

Items	Checkpoint	Value calculation of checkpoint	Total
Probability of function change	Change of core function	2.1	5.9
	Change of interface	3.2	
	Change of input	0.2	
	Change of output	0.4	
Analysis of quantitative of function completion	Identify importance of detailed function	3	4.6
	Identify number of detailed function	0.4	
	Quantitative of function completion	1.2	
Probability of function specification	Name of function	0.2	6.5
	Attribute of function	2.1	
	Operation of function	1.8	
	Relationship of other functions	2.4	

Table 6. Result of environment requirements

Items	Checkpoint	Value calculation of checkpoint	Total
Probability of transition	Identify of control command	1.8	2.32
	Probability of change of system architecture	2.8	
	Probability of change of system operation methodology	1.2	
Probability of change of relationship	Change of relationship of system components	1.8	3.3
	Disagreement of system components methodology	0.9	
	Disagreement of system components interface	2.8	

4) 세부항목의 비중에 입력 값을 곱하여 세부항목 간의 값을 합한다. 그리고 전체 체크포인트 항목의 값을 산출한다. 이때 체크포인트에 할당된 값을 초과할 수 없다.
5) 산출된 체크포인트 값을 다른 값들과 비교하여 우선순위를 정한다.
6) 우선순위에 의하여 대안을 만든다.

체크포인트의 연산방법에 의한 사례 적용은 4절에서 제시를 하였다.

4. 사례연구

3절에서 제시한 체크포인트를 활용하기 위하여 다음과 같은 분야를 선정하였다.

4.1 개발 배경

① 스마트폰 사용자가 점점 증가하면서 노년층과 유아층 사용자도 함께 증가하고 있다. 그러나 이와 같은 기능의 사용자 앱은 찾기 어렵다.
② 소외계층의 사용자에게 카카오톡, 네이버 등을 설치하고 추가적으로 사용할 수 있는 앱을 제작하고자 한다.

4.2 개발 목표

① 사전 준비 : 프로젝트 범위 및 세부일정 수립 (애자일 방식)
② 사전 준비 : 현재 앱 마켓에서 구할 수 있는 유아용 앱 분석
③ 안드로이드 기본학습 : 프로그래밍 경험
④ 제품 패키지 및 테스트 수행 : 실무적 품질 관리 체험

체크포인트를 적용하기 위한 개발내용은 제시한 바와 같다. 따라서 적용 분야를 3절에서 제시한 체크포인트의 기준과 절차에 의하여 중요도를 할당하였다.

사용자 요구사항의 체크포인트의 결과는 Table 5와 같다.

산출 결과에 의하면 기능 명세의 가능성에서 가장 높은 위험도를 내포하고 있다. 그 세부 내용으로는 “기능들과의 연관성”에서 가장 높은 위험도가 있었다. 따라서 위험도를 경감시키기 위하여 다음과 같은 대안을 만들었다.

① Play라는 클래스를 추가하여 액터에게 주는 메시지를 Play 클래스에게 전달
② OCR이라는 클래스를 추가하여 관련 메소드를 생성
③ ImageEdit 클래스를 추가하여 요구사항의 만족도를 향상시킴

환경 요구사항의 체크포인트의 결과는 Table 6과 같다. 산출 결과에 의하면 “상호 관계 변화 가능성”에서 가장 높은 위험도를 내포하고 있다. 그 세부 내용으로는 “시스템 구성 요소 간의 인터페이스 불일치”에서 가장 높은 위험도가 있었다. 따라서 위험도를 경감시키기 위하여 다음과 같은 대안을 만들었다.

① List 클래스가 존재하지 않기 때문에 getSelect(): List 메소드를 getSelect()로 수정
② getSelect() 메소드의 의미가 뚜렷하지 않아 Select()로 수정하여 선택된 대상의 여부를 판단하는 메소드로 의미를 부여

불확실성 및 가변성의 체크포인트의 결과는 Table 7과 같다.

산출 결과에 의하면 “가변성”에서 가장 높은 위험도를 내포하고 있다. 그 세부 내용으로는 “기능의 변경”에서 가장 높은 위험도가 있었다. 따라서 위험도를 경감시키기 위하여 다음과 같은 대안을 만들었다.

① 노인층, 유아층에서 요구하는 보정기능에 대한 클래스를 추가

Table 7. Result of uncertainly and variability

Items	Checkpoint	Value calculation of checkpoint	Total
Uncertainly	Ambiguity of function specification	3.2	3.4
	Ambiguity of data	1.8	
	Unpredictability of output	1.8	
Variability	Change of environment	1.4	3.55
	Change of system	0.6	
	Change of personal resource	0.3	
	Change of function	2.7	
	Change of relationship among functions	2.1	

② 노인층에서 요구하는 기능을 Revision 클래스의 하위 클래스 OCR을 생성

③ 유아층에서 요구하는 기능을 Revision 클래스의 하위 클래스 ImageEdit을 생성

사용자, 환경, 불확실성 및 가변성의 체크포인트 산출 결과에 의하여 문제점을 정량적으로 산출하였다. 또한 산출된 결과를 기반으로 하여 대안을 만들어서 문제점을 경감시키고자 했다.

5. 결론 및 향후 연구 방향

본 논문에서는 불확실성과 가변성을 내포하고 있는 개발 분야에서 문제점을 식별하고 해결하기 위하여 체크포인트를 제시하였다. 그리고 체크포인트를 정량적으로 산출하였으며, 이를 기반으로 대안을 제시하여 위험도를 경감시키고자 하였다.

향후 제시된 체크포인트에 대하여 중요도 부여 시, 참고할 수 있는 세분화된 기준을 설정하는 연구가 필요하다. 또한 웹과 연동하여 체크포인트를 활용할 수 있도록 도구를 개발할 필요가 있다.

References

[1] Barbee Teasley Mynatt, “Software Engineering with Student Project Guidance”, Prentice Hall, 1990.

[2] Axel van Lamsweerde, “Requirements Engineering”, Wiley, 2009.

[3] Yoon chung, “Successful Software development methodology”, Life power press, 1999.

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

[5] Choi eun man, “Software engineering”, Jungik publishing co., 2011.

[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, C. withrow, “Prediction and control of ada software defects, J. systems and software”, Vol.12, pp. 199-207, 1990.

[9] R. Balzer, N. Goldman, and D. Wile, “Informality in program specifications”, IEEE Transactions on Software Engineering, Vol.4, No.2, pp.94-103, 1978.

[10] L.A Zadeh, “Test-score semantic as a basis for computationa appoach to the representation of meaning”, Literacy Linguistic Computing, Vol.1, pp.24-35, 1986.

[11] L.A Zadeh, “Soft computing and fuzzy logic”, IEEE Software, Vol.11, No.6, pp.48-56, 1994.

[12] Shari Lawrence Pfleeger, “Basics of software engineering experimentation”, Kluwer Academic Publishers, Annex I, 2001.

이 은 서

e-mail : eslee@anu.ac.kr

2001년～현 재 ISO/IEC 15504 국제 선임 심사원

2004년 중앙대학교 컴퓨터공학과(박사)

2004년～현 재 임베디드 산업협회 전문위원

2004년～현 재 한국정보통신기술협회 위원

2012년～현 재 안동대학교 컴퓨터공학과 부교수

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