KIPS Trans. Softw. and Data Eng.
Vol.9, No.10 pp.297~302
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2020.9.10.297

A Study on Extraction of Additional Information Methodology for Defect Management in the Design Stage

Lee Eun-Ser†

ABSTRACT

Software design are an important phase in the software developments. In order to manage software design, we propose additional information. Additional information suggests a standard and quantitative methods. In this study, we propose additional information at the design stage for defect management.

Keywords : Defect Management, Quality of Design, Design of Critical Items

설계단계의 결함관리를 위한 추가정보 추출에 관한 연구

이 은 서†

요 약

소프트웨어 설계는 구현을 하기 위하여 중요한 단계이다. 소프트웨어 설계의 결함관리를 위하여 추가적인 정보를 제안하고자 한다. 추가적인 정보는 방법론을 제시하고 정량적인 방법으로 접근한다. 본 연구에서는 결함관리를 위하여 설계 단계의 추가적인 정보를 제시하고자 한다.

키워드 : 결함관리, 설계 품질, 결정적인 설계항목

1. 서 론

소프트웨어는 산업과 모든 생활에서 필수요소가 되고 있다. 하드웨어가 중요한 시기에는 전체 시스템에서 하드웨어 비용의 비중이 많이 차지하고 있으며 전체 시스템의 문제가 하드웨어에 집중되어 있었다[1-4]. 하드웨어의 높은 비중에서 소프트웨어가 많은 비중을 차지하면서 시스템에 문제의 원인이 소프트웨어가 많은 부분을 차지하게 되었다. 소프트웨어의 문제들은 오류, 결함, 버그, 실패와 같은 용어로 발생한다. 오류는 결함의 원인을 의미하며 버그는 만들어지는 소프트웨어에 포함된 결함이다. 실패는 소프트웨어를 만든 후에 실행 시에 발생되는 내용이다. 이와 같은 소프트웨어의 결함은 요구사항 분석한 결과를 기반으로 하여 설계를 수행하고 그 결과로 구현을 수행한다.

본 연구에서는 설계단계에서 수행할 때 결함을 관리하기 위하여 세 단계의 관점으로 접근하였다. 내용은 다음과 같다[5-9].

1. 요구사항의 분석내용을 설계에서 일관성 있게 수용하여 설계를 진행하는가?
2. 설계단계에서 사용자 관점에서 인터페이스를 위한 모듈, 컴포넌트, 프로세스의 입력과 출력 및 필요한 데이터가 명확하게 정의되어 있는가?
3. 설계단계의 산출물을 기반으로 일관성 있게 구현할 수 있도록 표준이 정의되고 있으며 이를 잘 수행하고 있는가?

설계단계에서 발생할 수 있는 결함의 원인을 찾아서 관리하고 제거하기 위한 정량적인 접근방법을 제시하고자 한다.

2. 기반 연구

2.1 설계단계의 결함

요구사항을 분석한 결과는 설계단계를 수행하기 위하여 기초 자료가 된다. 따라서 설계단계에서 요구사항의 분석결과를 일관성 있게 적용해야 하고 구현에 적합한 형태로 설계를 진행해야 한다. 이와 같은 내용이 적용되지 않는다면 전체적인 품질이 떨어진다. 그 내용은 다음과 같다[11-15].

1. 설계단계에서 완성해야 하는 필수적인 속성들
2. 요구사항을 일관성 있게 설계하는 것
3. 설계한 결과물에 대하여 완전성을 확인하는 것

※ 이 논문은 2019학년도 안동대학교 해외파견연구보조금에 의하여 연구되었음.
† 종신회원 : 안동대학교 컴퓨터공학과 교수
Manuscript Received : July 17, 2020
Accepted : July 30, 2020
* Corresponding Author : Lee Eun-Ser(eslee@anu.ac.kr)

4. 인터페이스를 통하여 모듈간의 통신과 연동을 완전하게 설계하는 것
5. 설계결과물의 원인인 요구사항을 추적할 수 있어야 하는 것

설계와 연관된 품질에서 결함을 관리하기 위하여 추가정보를 추출하여 설계의 결함관리를 수행하고자 한다.

2.2 소프트웨어 결함 관리

소프트웨어 결함 관리는 시스템의 품질을 결정하는데 중요한 요인이 된다. 결함 관리를 위하여 기본적으로 추적성을 기반으로 한다. 요구사항분석, 설계, 구현, 시험의 개발 단계가 서로 상호확인이 되어서 이전 단계의 결과물이 다음 단계에서 만들어진 부분을 확인할 수 있어야 한다. 설계단계에서도 소프트웨어 결함관리와 같이 진척상황 파악, 결함의 처리 상태와 결과를 프로세스화 하여 시스템에 적용해야 한다. 그리고 시스템에 적용하여 결함을 예방하여 발생의 원인을 제거하는 것이 중요한 목표이고 품질을 결정하는 요인이 된다. 또한 소프트웨어 개발을 수행하는 인적자원에 대하여 요구사항, 설계, 구현, 시험등을 수행하기 위한 기본 사항을 마련하고 주기적으로 교육을 수행하여 완전성 있고 정확한 소프트웨어가 만들어지도록 해야 한다[16-19].

3. 본 론

소프트웨어 개발 시에 결함을 줄이기 위하여 요구사항을 분석하고 설계, 구현, 시험 등의 개발과정에서 많은 기법과 방법론을 사용하고 있다. 설계 과정은 구현을 수행하기 전 단계로서 구현을 쉽고 정확하게 하기 위한 단계이다. 따라서 설계단계는 요구사항의 분석 내용을 기반으로 구현이 정확히 되도록 하기 위한 중간 단계이다. 설계단계에서 발생하는 결함유형은 다음과 같다[20].

1. 인터페이스를 활용하여 프로시저의 호출 유무와 필요한 파라미터 참조, 입출력 형식과 프인터페이스의 사용시 형태의 결함
2. 결함의 확인 시 오류를 표시하고 부적절한 방법과 내용을 확인하는 결함
3. 모듈이 실행되기 위한 데이터의 구조적인 문제와 내용에 관한 결함
4. 기능을 구현하기 위한 알고리즘과 포인터 사용유무, 반복문, 연산, 의도하는 기능의 수행과 연관된 결함
5. 시스템 : 구성, 시간, 메모리
6. 소프트웨어 구동하기 위한 환경의 정의와 설계, 구동을 위한 컴파일, 문제를 찾기 위한 시험, 다른 보조 시스템과의 연동과 관련된 결함

본 논문에서는 두 가지의 기준으로 설계의 결함관리를 위한 추가적인 정보를 제공하고자 한다. 기준은 다음과 같다.

1. 설계하려는 요소들이 요구사항과의 일관성과 정확성을 유지하면서 수행되는지를 확인한다.
2. 설계가 진행되면서 산출되는 명세들이 구현관점에서 만들어지는지 확인한다.

결함관리를 위한 추가적인 정보를 제공하기 위하여 두 가지 기준을 기반으로 다음과 같은 사항을 확인하고자 한다.

1. 요구사항의 분석내용을 설계에서 일관성 있게 수용하여 설계를 진행하는가?
2. 설계단계에서 사용자 관점에서 인터페이스를 위한 모듈, 컴포넌트, 프로세스의 입력과 출력 및 필요한 데이터가 명확하게 정의되어 있는가?
3. 설계단계의 산출물을 기반으로 일관성 있게 구현할 수 있도록 표준이 정의되고 있으며 이를 잘 수행하고 있는가?

본 논문에서는 위의 세 가지 사항을 기반으로 결함관리를 위한 추가적인 정보를 제공하고자 한다. 추가적인 정보는 결함을 예방하고 발생된 결함의 원인과 위치를 신속하게 찾아서 제거하는데 활용될 수 있다. 이를 기반으로 하여 설계를 수행하기 전에 결함을 예측하여 미리 대비하는데에도 활용될 수 있다. 설계단계의 추가적인 정보를 DCI (Design of Critical Items)라고 정의하였다. DCI를 정의하기 위하여 다음과 같은 결함관리 방안에 의하여 수행된다. 도식화된 형태는 다음과 같다.

Fig. 1에서는 DCI의 프로세스를 제시하고 있다. DCI 프로세스는 요구사항의 분석내용을 설계에 적용하기 위하여 이를 정의하고 적용한다. 또한 설계에 잘 적용되었는지 확인을 한다. 적용이 되지 않은 경우에는 DCI 프로세스를 지속적으로 반복하여 이전 단계의 항목이 누락되거나 잘못 적용되지 않도록 관리한다.

DCI에서의 결함은 이전 단계에서 분석된 내용의 적용과 이를 구현으로 진행하기 위하여 문제가 되는 내용이 결함에 해당된다. 이를 찾아서 원인을 제거하여 소프트웨어를 개발하는 동안 결함을 관리하여 전체 시스템의 신뢰성을 구축하고자 한다.

DCI에서 결함의 원인을 찾아서 제거하기 위해서는 결함여부에 대한 판단 기준이 필요하다. 결함의 판단 기준은 결함의 정도를 파악하기 위한 것으로서 다음과 같이 제시한다.

Fig. 1. Process of DCI

Table 1. Criteria of Degree of Defect

Measurement	Points
H (High)	80 or more
N (Normal)	51-79
L (Low)	0-49

결함의 판단 기준은 DCI 기준으로 H는 요구사항의 분석 내용을 잘 반영하면서 구현을 고려하는 경우이다. N은 설계의 내용이 구현을 고려하지 않아서 구현 시 설계의 내용을 다시 검토하거나 수정하는 경우이다. L은 요구사항의 분석내용을 반영하지도 못하고 구현을 고려해서 설계를 수행하진 않는 경우이다. DCI에 의한 판단 기준을 여러 가지 관점에 의하여 판단할 수 있지만 본 논문에서는 전체 시스템의 완성도를 관리하기 위하여 이전 단계의 요구사항 내용 반영과 구현을 정확하고 쉽게 할 수 있는지에 초점을 맞추었다.

3.1 DCI(Design of Critical Items)

설계단계에서 발생되는 결함을 확인하여 결함 여부를 판단하고 관리하기 위하여 추가적인 정보가 필요하게 된다. 추가적인 정보는 다음의 기준으로 산출한다.

1. 요구사항의 분석내용을 설계에서 일관성 있게 수용하여 설계를 진행하는가?
2. 설계단계에서 사용자 관점에서 인터페이스를 위한 모듈, 컴포넌트, 프로세스의 입력과 출력 및 필요한 데이터가 명확하게 정의되어 있는가?
3. 설계단계의 산출물을 기반으로 일관성 있게 구현할 수 있도록 표준이 정의되고 있으며 이를 잘 수행하고 있는가?

1(요구사항의 분석내용을 설계에서 일관성 있게 수용하여 설계를 진행하는가?)의 경우 요구사항 분석의 내용을 설계에 적용 여부를 확인하기 위하여 추적성을 활용하였다. 추적성을 평가하기 위하여 3가지 항목에 의하여 판단 기준을 제시하였다. 3가지 항목은 적용할 요구사항 분석 확인, 변환 알고리즘 또는 방법, 적용 여부 확인으로 구성하였다. 판단 기준의 내용은 다음과 같다.

Table 2. Criteria of Traceability

	Contents
Confirm requirements analysis to apply	Confirm the content of requirement analysis to be applied to the design
Transformation algorithm or method	The presence of design methodologies or transformation algorithms
Check for applicability	Confirmation of requirements analysis applied to design

적용할 요구사항 분석 확인은 요구사항 분석의 결과물이 명확히 확인되는가를 판단하는 것이다. 변환 알고리즘 또는 방법은 요구사항 분석에서 추출된 항목을 설계에 적용하기 위하여 방법론이나 알고리즘이 존재하는가를 판단하는 것이다. 적용 여부 확인은 요구사항 분석에서 추출된 항목을 설계에 적용한 것이 확인되는지를 판단하는 것이다.

추적성의 성숙도는 3단계로 제시하였다. 추적성의 성숙도를 구분하기 위하여 DT(Degree of traceability)로 명명하였다. 성숙도는 상, 중, 하로 구분하였다. 이를 기반으로 하여 높음, 중간, 낮음의 3단계로 성숙도를 제시하였다. DT-H는 적용할 요구사항 분석 확인, 변환 알고리즘 또는 방법, 적용 여부 확인이 명확히 확인되는 경우이다. DT-N은 적용할 요구사항 분석 확인, 변환 알고리즘 또는 방법, 적용 여부 확인 중에서 2개 항목이 확인되는 경우이다. DT-L은 적용할 요구사항 분석 확인, 변환 알고리즘 또는 방법, 적용 여부 확인 중에서 1개 이하만 확인되는 경우이다. 가중치는 설계 산출물에 따라 차등 적용하게 된다. 추적성에 대한 성숙도는 다음과 같이 제시하였다.

Table 3. Degree of Traceability

Measurement	Points
DT-H (Degree of Traceability-High)	80 or more
DT-N (Degree of Traceability-Normal)	50-79
DT-L (Degree of Traceability-Low)	0-49

2(설계단계에서 사용자 관점에서 인터페이스를 위한 모듈, 컴포넌트, 프로세스의 입력과 출력 및 필요한 데이터가 명확하게 정의되어 있는가?)의 경우 설계단계에서 사용자 관점이 구현관점에서 잘 정의되어서 요구사항이 구현으로 정의되는지를 평가한다. 이를 판단하기 위하여 다음과 같은 기준을 사용한다.

Table 4. Criteria of Design View of user Interface

	Contents
Interface Extraction	Extraction of Interfaces for Function Execution
Definition of Interface Parameters	Define data required for interface execution
Definition of Processes	Define steps and processes to run the interface

인터페이스 추출은 기능 수행을 위하여 사용되는 인터페이스가 추출되어 있는지를 판단한다. 인터페이스 파라미터 자료정의는 인터페이스 수행을 위하여 필요로 하는 데이터가 구분되고 정의되어 있는지를 판단한다. 프로세스 정의는 인터페이스가 실행 시에 해당되는 단계와 프로세스가 명확히 정의되어 구분되는지를 판단한다.

DI-H는 인터페이스 추출을 수행하고 있으며 인터페이스 실행 시, 사용되는 파라미터와 대상 프로세스가 정의되고 분석되는 경우이다. DI-N은 인터페이스 추출을 수행할 때, 인터페이스가 사용하는 파라미터와 자료를 같이 추출하는지를 평가한다. DI-L은 인터페이스 추출과 실행 시 요구되는 파라미터 추출, 대상 프로세스의 분석이 수행되지 않는 경우이다. 이를 판단하기 위하여 다음과 같은 기준을 사용한다.

Table 5. Degree of Traceability

Measurement	Points
DI-H (Degree of Interface-High)	80 or more
DIP-N (Degree of Interface-Normal)	51-79
DIP-L (Degree of Interface-Low)	0-50

3(설계단계의 산출물을 기반으로 일관성 있게 구현할 수 있도록 표준이 정의되고 있으며 이를 잘 수행하고 있는가?)의 경우 설계 시에 표준을 가지고 있으며 이를 기반하여 수행하는가를 판단한다. 이를 판단하기 위하여 다음과 같은 기준을 사용한다.

Table 6. Criteria of Design Standard

	Contents
Design standard	Building design standards
Interface standard	Compliant with interface standards
Extraction of reuse unit	Extract units for reuse (Object, Class, Function etc.)

인터페이스 관점의 설계 표준 성숙도를 측정하기 위하여 3단계로 제시하였다. 인터페이스 관점의 설계 성숙도를 구분하기 위하여 DI(Degree of Interface)로 명명하였다. 성숙도는 상, 중, 하로 구분하였다. 이를 기반으로 하여 높음, 중간, 낮음의 3단계로 성숙도를 제시하였다. DIS-H는 인터페이스 생성을 표준을 준수하고 있으며 전체 재사용 단위 중에서 80% 이상인 경우이다. DIS-N은 인터페이스 생성 시, 표준을 준수하는 비율이 50-79%가 되는 경우이다. DIS-L은 인터페이스를 추출하기 위한 설계 및 활동이 없으며 인터페이스로 연결이 되지 않고 인터페이스를 사용하더라도 그 범위가 전체 재사용 단위 중에서 50% 미만인 경우이다. 이를 판단하기 위하여 다음과 같은 기준을 사용한다.

설계 표준은 설계를 진행하는 주체가 표준에 의하여 설계 과정을 진행하는가를 판단한다. 인터페이스 표준은 설계의 구성요소가 인터페이스를 연결하는 표준 방법에 의하여 연결이 되어 있는가를 판단한다. 재사용 단위 추출은 설계에서 추출한 기능들이 재사용을 하기 위한 단위로 추출되어 있는지를 판단한다. 그 단위는 객체, 클래스, 함수 등이 해당된다.

Table 7. Degree of Interface

Measurement	Points
DIS-H (Degree of Interface Standard-High)	80 or more
DIS-N (Degree of Interface Standard-Normal)	51-79
DIS-L (Degree of Interface Standard-Low)	0-50

DCI를 기반으로 설계단계의 성숙도를 측정하고자 한다. DCI 성숙도는 DDCI (Degree of Design of Critical Items)로 정의하였으며 측정하기 위하여 다음 수식을 활용한다.

$$(DT(\sum_{DT=1}^{i}(DTpoint * Weight)) + (DI(\sum_{DIE=1}^{i}(DIpoint * Weight)) + (DIS(\sum_{DIS=1}^{i}(DISpoint * Weight))$$

각 단계별 수식에 의하여 산출하게 된다. 산출된 결과를 기반으로 하여 성숙도는 3단계로 제시하고자 한다. 결과값은 70이상이 결함 발생 가능성이 높고, 41-69인 경우에는 중간, 40이하인 경우에는 발생 가능성이 낮게 나오는 것으로 기준을 정하였다.

Table 8. Degree of Design of Critical Items

Measurement	Contents
DDCI-H (Performance-High)	Result value of 70 or higher
DDCI-N (Performance-Normal)	Results value from 41 to 69
DDCI-L (Performance-Low)	Result value less than 40

4. 적용 및 사례

본 연구에서는 설계단계에서 발생되는 결함을 확인하기 위하여 DCI를 제시하였다. 이와 같은 이론을 개발 내용에 적용하여 효과를 확인하고자 한다. 설계의 요소를 측정하기 위하여 요구사항이 설계로 연결이 되는지를 판단하기 위하여 추적성을 분석하여 정의하였다.

Table 9. Traceability

Requirements		설계						
Requirements name	Usecase Diagram	Class Diagram		Sequence Diagram		State Diagram		
	Usecase name	System	Class name	Sequence name	Correspondence class	State	State name	Correspondence sepuence
0001 map check	0001 map check	application system	UserControlManager(user control)	fixed location setting	UserControlManager	App	loading screen	0
0002 Show location	0002 Show location		MapController(implement)	fixed location check	FixedPositionController, ClientSocketManager		login screen	login sequence
0003 location navigation	0003 location navigation	application system	ClientSocketManager(reception)	map view	UserControlManager, MapController		join membership scre	join membership sequence
		control system	ServerSocketManager(Transmission)	Communication switching	ClientSocketManager		menu screen	0
			PositionGetter(implement)	check the scenery	UserControlManager, ClientSocketManager		signal sending	notify protector sequence
0004 Communication switching	0004 Communication switching	control system	NetworkController	Protected person manual	ServerSocketManager, ClientSocketHandler, UserControlManager, ButtonListner		fixed location setting	fixed location setting sequence
			NetworkController	join membership	ServerSocketManager, ClientSocketHandler, UserControlManager		check the scenery	check the scnery sequence
0005 notify protector	0005 notify protector	control system	ServerSocketManager	login	LoginActivity		Protected person loc	map view sequence
			ButtonListner		SignupActvity	Button	Button recognition	Protected person notify sequence
0006 Protected person manual notify	0006 Protected person manual notify	application system	ClientSocketHandler				signal sending	Protected person notify sequence
0007 notify protected person	0007 notify protected person	control system	ServerSocketManager			Server	Wait for application s	0
0008 Notification signalling	0008 Notification signalling	–					signal analysis	check the scnery, protector notify, fixed location setting, fixed location check, map view
0009 Location fixing settings	0009 Location fixing settings	application system	ClientSocketManager				Take and send pictur	check the scnery
			UserControlManager(user control)				location information tr	map view
			FixedPositionController(implement)				Signal reception	protector notify
		control system	ServerSocketManager(reception)					
			PositionGetter					
0010 fixed location Breakaway Whether check	010 fixed location Breakaway Whether chec	application system	ClientSocketManager					
			FixedPositionController					
0011 fixed location Breakaway notify	0011 fixed location Breakaway notify	application system	FixedPositionController					
0012 Check the surrounding scenery	0012 Check the surrounding scenery	application system	ClientSocketHandler					
			UserControlManager					
0013 Camera operating signal transmission	0013 Camera operating signal transmission	application system	ClientSocketManager					
0014 Take a picture	0014 Take a picture	control system	ServerSocketManager					
			CameraController					
0015 join membership	0015 join membership	application system	SignupActivity					
0016 login	0016 login	application system	LoginActivity					
0017 application button click	0017 application button click	application system	event function					
0018 device button click	0018 device button click	control system	ButtonListner					
0019 photo transmission	0019 photo transmission	application system	ClientSocketManager					
		control system	ServerSocketManager					
0020 notify	–							
0021 Device on/off	–							

Table 10. Result

Main Category	Serial number	DT		DI		DIS		Result
		Points	Weight	Points	Weight	Points	Weight	
Function	0001	60	0.5	90	0.2	80	0.3	72
	0002	60	0.5	60	0.2	50	0.3	57
	0003	70	0.5	30	0.3	70	0.2	58
	0004	60	0.5	50	0.2	60	0.3	58
	0005	70	0.6	60	0.2	60	0.2	66
	0006	80	0.2	80	0.6	90	0.2	82
	0007	80	0.2	80	0.6	90	0.2	82
	0008	70	0.6	80	0.2	70	0.2	72
	0009	90	0.3	90	0.2	60	0.5	75
	0010	70	0.4	70	0.3	70	0.3	70
	0011	40	0.5	50	0.3	60	0.2	47
	0012	50	0.5	30	0.3	20	0.2	38
	0013	40	0.4	40	0.4	40	0.2	40
	0014	80	0.5	80	0.3	80	0.2	80
	0015	90	0.3	90	0.4	90	0.3	90
	0016	90	0.3	90	0.4	90	0.3	90
	0017	90	0.3	90	0.4	90	0.3	90
	0018	90	0.3	90	0.4	90	0.3	90
	0019	70	0.3	80	0.4	60	0.3	71
	0020	90	0.3	90	0.4	90	0.3	90
	0021	90	0.3	90	0.4	90	0.3	90
Non-function	N-0020	90	0.1	90	0.2	90	0.7	90
	N-0021	90	0.1	90	0.2	90	0.7	90
	N-0022	90	0.1	90	0.2	90	0.7	90
	N-0023	90	0.1	90	0.2	90	0.7	90

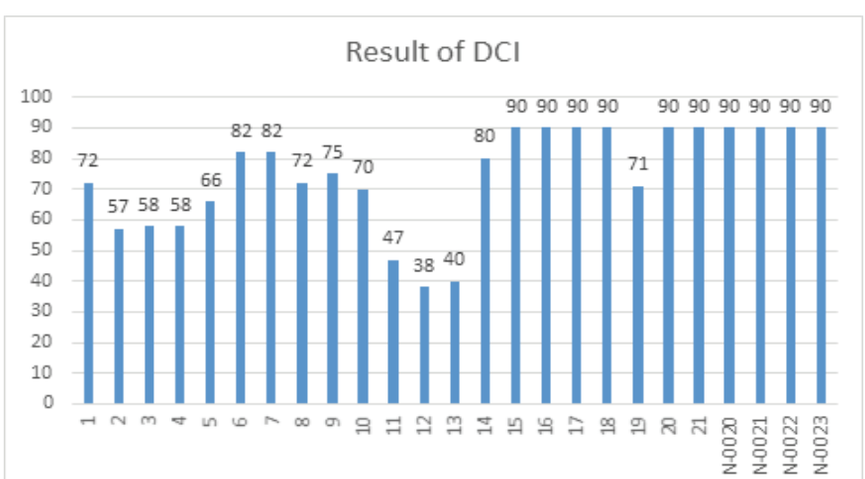

Fig. 2. Process of DCI

추적성을 기반으로 요구사항이 잘 정의되고 구현을 위하여 설계를 잘 정의하는지를 확인하고자 한다.

각 항목의 Points와 Weight를 기반으로 Result를 산정하였다. 산정한 결과는 다음과 같이 그래프로 표현하였다.

세부 산정 결과에 따르면 2, 3, 4, 5, 11번 항목의 결함 발생률이 중간으로 산정되었다. 그리고 12, 13번 항목이 결함 발생률이 높은 것으로 산정되었다. 산정된 결과를 기반으로 12, 13번 항목을 중점적으로 결함을 관리할 필요가 있다. 또한 2, 3, 4, 5, 11번 항목도 결함으로 진행되지 않도록 관리할 필요가 있다.

5. 결 론

본 연구에서는 요구사항의 추출 요소가 설계에서 잘 정의되고 구현에서 실현되는 신뢰성과 정확도를 높이기 위하여 설계 관점에서 결함 발생 가능성을 찾고자 한다. 그 기준은 요구사항의 분석과 적용 관점, 인터페이스 추출 관점, 인터페이스 생성 시에 표준 관점으로 제시하였다.

향후 연구내용으로는 요구사항과 설계, 구현의 결함 발생 가능성을 제시하여 전체적으로 통합하는 과정이 필요하다. 통합을 통하여 전체 시스템의 결함 발생 가능성을 정량적으로 산정하여 결함 관리를 하고자 한다.

References

[1] Roger S. Pressman, "Software Engineering," McGraw-Hill Higher Education, 2005.

[2] Harold W. Lawson, "An Assessment Methodology for Safety Critical Computer Based Systems," *Proceedings of CSR 12th Annual Workshop on Safety and Reliability of Software Based Systems*, pp.183-200, 1995.

[3] Peter L. Goddard, "A Combines Analysis Approach To Assessing Requirements for Safety Critical Real-Time Control Systems," *Proceeding Annual Reliability & Maintainability Symposium*, pp.227-230, 1993.

[4] Swarup, M. Ben and P. Seetha Ramaiah, "A Software Safety Model for Safety Critical Application," *Proceedings of International Forum on Computer Science-Technology and Applications*, pp.21-32, 2009.

[5] Stephen S. Cha, Nancy G. Leveson, and Timothy J. Shimeall, "Safety Verification in MURPHY using Fault Tree Analysis," *Proceeding on the 10th International Conference on Software Engineering*, pp.377-386, 1988.

[6] Hye-Jung Jung, "The Analysis of Software Fault and Application Method of Weight using the Testing Data," *Journal of Korea Multimedia Society*, Vol.14, No.6, pp. 766-774, 2011.

[7] IEC 60812, "Procedures for Failure Mode and Effect Analysis (FMEA)," 2006.

[8] SEMATECH, "Failure Modes and Effects Analysis (FMEA): A Guide for Continuous Improvement for the Semiconductor Equipment Industry," 1992.

[9] IEEE Standard for Software Reviews, IEEE Std 1028-2008, 1998.

[10] Software Engineering Standards, IEEE Computer Society, 1994.

[11] I. Somerville, Software Engineering, Addison-Wesley, 2001.

[12] "Software Risk Abatement", AFCS/AFLC Pamphlet 800-45, U.S. Air Force, 1988.

[13] B, W. Boehm, "Software Risk Management", IEEE Computer Society Press, 1989.

[14] D. W. Karolak, "Software Engineering Risk Management", IEEE Computer Society Pess, 1996.

[15] "Software bugs cost US economy dear," Jun. 10, 2009. Archived from the original on June 10, 2009. Retrieved Sept. 24, 2012.

[16] IEEE Annals of the History of Computing, Vol.22, Issue.1, 2000.

[17] "Testing experience : te : the magazine for professional testers," Testing Experience. Germany: testingexperience: 42, Mar. 2012.

[18] McDonald, Marc; Musson, Robert; Smith, Ross. The Practical Guide to Defect Prevention. Microsoft Press. p. 480, 2007.

[19] Allen, Mitch (May-June 2002). "Bug Tracking Basics: A beginner's guide to reporting and tracking defects," Software Testing & Quality Engineering Magazine, Vol.4 No.3. pp.20-24. Retrieved Dece. 19, 2017.

[20] Watts S. Humphrey, "Introduction to the Personal Software Process," PEASON, 2003.

이 은 서

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

e-mail : eslee@anu.ac.kr

2001 ~ 현 재 ISO/IEC 15504 국제 선임 심사원

2004년 중앙대학교 컴퓨터공학과(박사)

2008년 ~ 현 재 안동대학교 컴퓨터공학과 교수

관심분야 : CBD, Formal Method, Quality Model, SPI(Defect Analysis)