안전-필수 소프트웨어를 위한 신뢰도(Dependability) 프로세스에 관한 연구

김영미 ° 정충희
한국원자력안전기술원 공학기준개발실
ymkim@kins.re.kr, chjeong@kins.re.kr

A Study on the Dependability Processes for Safety Critical Software

Youngmi Kim° Choongheui Jeong
Korea Institute of Nuclear Safety

요 약

최근 디지털 컴퓨터와 정보처리기술의 발전과 더불어 원자력 발전소의 계측제어시스템과 같은 안전-필수 시스템에서도 디지털 기술을 채택하기 시작했다. 안전-필수 시스템에 사용되는 소프트웨어는 높은 신뢰도(dependability)가 요구된다. 소프트웨어의 신뢰도는 신뢰성(reliability), 안전성, 보안 등 다양한 속성들로 설명될 수 있다. 소프트웨어의 신뢰도 향상을 위한 프로세스는 결함예방프로세스, 결함허용프로세스, 결함제거프로세스 그리고 결함예측프로세스가 있으며 이들 프로세스는 소프트웨어 수명주기 초반부터 수행되어야 한다. 본 논문에서는 소프트웨어 신뢰도향상을 위한 신뢰도 프로세스 모델과 개발 단계별로 수행되어야 할 신뢰도 태스크를 제시한다.

1. 서 론

최근 디지털 기술의 발전과 함께 원전 계측제어 시스템과 같이 안전-필수 기능이 요구되는 곳에도 디지털 기술들이 도입되기 시작했다. 이와 함께 안전-필수 시스템에서 사용되는 소프트웨어는 높은 신뢰도를 요구하게 되었다. 안전-필수 소프트웨어가 잘못 수행되거나 혹은 적절히 수행되지 못하는 경우는 사람 혹은 주위 환경에 직접적으로 해를 입힐 수 있으며 심지어는 사람의 생명까지도 앗아갈 수 있다. 고신뢰도, 고 안전성이 요구되는 소프트웨어가 가져야 될 특성이 바로 신뢰도(dependability)이다. 특히, 국방, 항공, 원자력 등의 산업분야에서 소프트웨어의 신뢰도에 대한 연구를 계속 해오고 있다. 소프트웨어의 신뢰도는 기본적으로 신뢰성(reliability), 안전성, 가용성, 무결성, 기밀성, 유지보수성 등과 같은 속성을 가진다. 높은 신뢰도를 가진 소프트웨어를 개발하기 위해서는 신뢰도의 속성들이 소프트웨어 초기 단계부터 체계적으로 고려되어야 한다.

본 논문에서는 소프트웨어의 신뢰도 향상을 위해 소프트웨어의 수명주기에 소프트웨어 신뢰도 프로세스를 포함시키는 신뢰도 프로세스 모델과 각 개발 단계별로 수행되어야 하는 신뢰도 태스크를 제시한다.

제 2장에서는 본 연구와 관련된 소프트웨어 신뢰도 및 신뢰도프로세스와 관련된 연구배경을 소개하고, 제 3장에서는 소프트웨어수명주기와 신뢰도 프로세스를 통합하기 위한 신뢰도 모델을 제시하고 제 4장에서는 각 개발 단계별 신뢰도 태스크를 제시한다. 그리고, 제 5장에서 결론을 맺는다.

2. 연구배경

2.1 소프트웨어의 신뢰도(Dependability)

소프트웨어의 신뢰도는 주로 서로 다른 여러 가지 속성들을 포함하는 통합적인 의미로 사용된다. 컴퓨터 시스템의 신뢰도란 시스템이 신뢰할 수 있는 서비스를 제공할 능력을 말한다[1].

그림 1은 소프트웨어 신뢰도의 전반적인 특성을 보여준다. 소프트웨어 신뢰도의 속성은 사용자의 요구사항에 따라 다양하게 정의될 수 있으며, 기본적으로 신뢰성, 안전성, 가용성, 무결성 그리고 기밀성 등을 들 수 있다. 신뢰도는 결함예방, 결함허용, 결함제거 그리고 결함예측 등의 방법을 통해 향상될 수 있으며, 결함 및 실패, 위협(위험을 초래하는 것들), 공격, 업그레이드 등에 의해 손상을 받는다. 또한, 신뢰도의 측정은 대상항목의 특성에 따라 MTTF, 발생확률, 발생 횟수 등으로 가능하다[2][3].

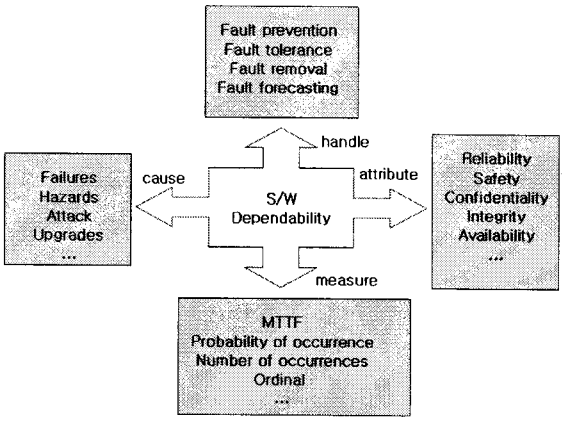

그림 1 S/W 신뢰도(dependability)의 전반적인 특성

2.2 소프트웨어 신뢰도(Dependability)의 속성

소프트웨어의 신뢰도를 구성하는 속성들은 시스템의 이해당사자들의 요구에 따라 달라질 수 있다. 기본적인 속성들로는 신뢰성(reliability), 안전성, 기밀성, 무결성, 가용성, 유지보수성 등이 있다. 기밀성, 무결성, 가용성 속성은 함께 묶어 보안특성으로 부르기도 한다. 각 속성에 대한 정의는 정의하는 주체에 따라 조금씩 달라질 수 있다. 다음은 그 정의들 중의 하나이다.

- 신뢰성 : 정확한 서비스를 지속적으로 제공할 능력 [4]
- 안전성 : 재앙을 유발할 수 있는 결과를 발생시키지 않을 능력(죽음, 부상, 직업병, 자산의 피해, 환경손상 등을 유발하지 않을 능력) [4]
- 보안 : 허가 받지 않은 노출없이(기밀성), 정보의 변경없이(무결성), 적법한 사용자에게는 서비스 거부없이 요구되는 서비스를 제공할 능력(가용성) [5]

소프트웨어의 신뢰도 속성들에 대하여 여러 문헌에서 많은 정의들을 내리고 있지만 대부분 개괄적이고 정성적이다. 또한, 신뢰도 속성들은 서로 중첩되거나 상충되는 특성들도 가지고 있다. 이러한 특성들은 소프트웨어 수명주기 초반에 이해당사자들의 합의하에 보완되고 조정되어야 한다.

2.3 소프트웨어 신뢰도(Dependability)의 기본 프로세스

소프트웨어 신뢰도 향상을 위해 사용되는 프로세스에는 결함예방프로세스, 결함허용프로세스, 결함제거프로세스 그리고 결함예측프로세스가 있다. 결함예방프로세스는 결함의 발생을 어떻게 막을 것인가에 초점을 맞추고, 결함포용프로세스는 결함이 존재하고 있는 상황하에서 어떻게 정확한 서비스를 제공할 것인가에 초점을 맞춘다. 결함제거프로세스는 결함의 수나 심각성을 어떻게 낮출 수 있느냐에 초점을 맞추며, 결함 예측프로세스는 현재 존재하는 결함의 수나 미래의 사건, 그리고, 결함으로 야기되는 결과에 대한 예측에 초점을 맞춘다.

3 신뢰도(dependability) 프로세스 모델

소프트웨어 수명주기와 관련된 주요 국제표준으로는 ISO/IEC12207과 IEEE Std. 1074가 있다. ISO/IEC12207은 소프트웨어 수명주기 모형을 제시하고 있으며, IEEE Std. 1074는 소프트웨어 수명주기를 구현하기 위한 구체적인 방법을 제시하고 있다.

IEEE Std. 1012에서는 ISO/IEC12207의 전 수명주기 동안의 확인 및 검증활동에 대해 제시하고 있다[6]. 소프트웨어 수명주기 동안의 확인 및 검증활동은 소프트웨어의 신뢰도 향상을 위한 많은 활동들을 포함하고 있다. IEEE Std. 1012에서는 표 1과 같이 각 소프트웨어에 대해 무결성수준을 정의하고 이에 따라 차등화된 확인 및 검증을 실시하도록 하고 있다 [6].

안전-필수 서비스를 제공하는 고신뢰도 시스템을 설계하기 위해서는 확인 및 검증활동 이외에 신뢰도프로세스가 명시적으로 수행되어야 한다 [3]. 그림 2는 소프트웨어 기본수명주기프로세스와 신뢰도 프로세스들을 보여준다. 결함예방, 결함허용, 결함제거, 결함예측 프로세스와 같은 신뢰도 프로세스는 소프트웨어 전 수명주기 동안 같이 이루어져야 한다.

본 연구에서는 소프트웨어수명주기 가운데 기본 공정 중에서 개발공정 즉 개념, 요구사항, 설계, 구현, 시험, 설치 및 정검단계에 초점을 맞추었다.

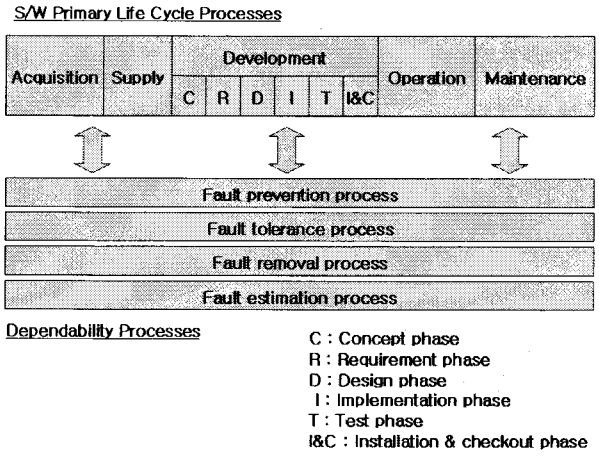

그림 2. S/W 개발 및 신뢰도 프로세스 모델

4 소프트웨어 개발을 위한 신뢰도(dependability) 태스크

그림 2에서와 같이 소프트웨어 개발을 위한 기본 단계는 개념 단계, 요구사항 단계, 설계 단계, 구현 단계, 테스트 단계, 그리고 설치 및 점검 단계로 구성되어 있다. 본 논문에서는 각 소프트웨어 개발 단계마다 신뢰성, 안전성, 보안을 향상시키기 위해 수행해야 하는 신뢰도 태스크의 목록을 제시한다. 안전-필수 소프트웨어를 위한 신뢰도프로세스는 소프트웨어 수명주기 초기 단계에서부터 수행되어야 한다. 특히 결함예방 프로세스의 경우는 많은 태스크가 요구사항 단계에서 이루어진다. 또한, 결함예방을 위해 수행되는 많은 태스크는 결함허용프로세스, 결함제거프로세스, 결함예측프로세스를 수행하기 위해 필요한 정보들을 제공해준다. 본 논문에서는 개념 단계 및 요구사항 단계에 수행하는 신뢰도 속성별 신뢰도 태스크를 제시하였다.

표 1. 소프트웨어의 무결성수준[6]

설 명	무결성수준
소프트웨어가 정확하게 수행되지 않으면 심각한 결과(죽음, 시스템 파괴, 경제적 사회적 손실)를 초래하며 완화될 수 없음.	1
소프트웨어가 정확하게 수행되지 않으면 심각한 결과(영구적 부상, 주요 시스템 기능 악화, 경제적 사회적 영향)을 초래하며 부분적으로 완화될 수 있음.	2
소프트웨어가 정확하게 수행되지 않으면 작은 문제가 발생할 수 있으며 완전히 완화가 가능함.	3
소프트웨어가 정확하게 수행되지 않아도 무시할 수 있을 만한 문제가 생김. 완화가 필요 없음.	4

4.1 결함예방 및 결함제거프로세스와 소프트웨어개발

결함예방프로세스는 결함의 발생을 사전에 예방하는 것을 목표로 하며, 개발계획 작성, 조직구성, 개발 표준 및 개발 도구의 선택 등과 깊은 관련이 있다. 이러한 활동들은 소프트웨어개발의 초기 단계에 결정되며 소프트웨어 수명주기 동안 지속적으로 영향을 미치게 된다. 표 2는 소프트웨어 개발공정 동안에 수행되어야 하는 대표적인 결함예방태스크를 보여준다.

개념 단계에서 신뢰성을 향상시키기 위해 수행해야 하는 주요 결함예방프로세스는 사용자의 요구에 맞도록 개념문서가 작성되었는지 확인하는 것이다. 안전성 측면에서는 각 소프트웨어의 무결성수준을 결정하고 잠재적인 위해도 및 기술적, 관리적 위험을 확인하고 위험도 분석을 수행하여 결함 및 위해도 목록을 작성한다. 보안 측면에서는 보안 위험도에 대하여 사용자가 허용 가능한 수준을 확인한다.

요구사항 단계에서는 신뢰성 향상을 위해 이전 단계와 다음 단계간에 해당항목 간의 양방향 추적이 가능해야 하며 각 단계 산출물의 정확성, 일관성, 완전성이 분석되어야 한다. 또한, 안전성 측면에서 각 단계마다 위험도 분석을 수행하여 새로운 위험발생 여부를 평가해야 한다. 보안측면에서는 보안 요구사항들이 보안위험을 허용 가능한 수준으로 낮춰줌을 확인해야 한다.

표 2. 결함예방 태스크와 소프트웨어 수명주기

개발 단계	결함예방 태스크
개념	*신뢰성* ● 개념문서와 사용자의 요구사항 일치여부 확인 ● 시스템 요구사항 분석 *안전성* ● 소프트웨어 무결성수준 결정 ● 개념적인 시스템으로부터 잠재적 위해도 확인 ● 기술적, 관리적 위험 확인, 위험도 분석 수행 *보안* ● 보안 위험도에 대한 사용자의 허용 가능한 수준 확인
요구사항	*신뢰성* ● 개념에 대비한 요구사항의 정확성, 일관성, 완전성 분석 *안전성* ● 결정된 소프트웨어 무결성수준 재검토 ● 시스템의 위해도에 영향을 미치는 요구사항 확인 ● 위험도 분석 검토 및 수정 *보안* ● 개념단계에서 확인된 보안위험에 대응하는 시스템 보안 요구사항 확인
설계	*신뢰성* ● 설계와 요구사항간의 양방향 추적성 분석 ● 설계의 정확성, 일관성, 완전성 분석 ● 인터페이스 분석 *안전성* ● 결정된 소프트웨어 무결성수준 재검토 ● 로직설계 및 데이터 구성요소가 필수 요구사항을 정확히 구현했는지 확인 ● 위해도 및 위험도 분석 검토 및 수정 *보안* ● 구조 및 상세설계가 보안요구사항을 만족하는지 확인 ● 외부 컴포넌트와의 인터페이스로 야기되는 위험 확인
구현	*신뢰성*

	● 소스코드와 설계간의 양방향 추적성 분석 ● 소스코드의 정확성, 일관성, 완전성 분석 ● 인터페이스 분석 *안전성* ● 결정된 소프트웨어 무결성수준 재검토 ● 구현 및 데이터 구성요소가 필수 요구사항을 정확히 구현했는지 확인 ● 위해도 및 위험도 분석 검토 및 수정 *보안* ● 보안 요구사항이 올바르게 구현되었는지 확인 ● 새로운 보안 위험이 발생하지 않았는지 확인
테스트	*신뢰성* ● 테스트 계획, 설계, 사례 및 절차의 정확성, 완전성 분석 *안전성* ● 테스트 도구의 위험 야기 여부확인 ● 위험도 분석 검토 및 수정 *보안* ● 구현된 시스템의 보안위험 증가여부 확인
설치 및 점검	*신뢰성* ● 설치되어야 할 모든 소프트웨어 확인 ● 현장에 맞는 변수 및 상황 확인 ● 전환 시 지속적인 운영 및 서비스 제공을 위한 요구사항 확인 *안전성* ● 설치 절차 및 환경으로 야기되는 위험 확인 ● 위해도 및 위험도 분석 검토 및 수정 *보안* ● 설치된 소프트웨어에 새로운 혹은 증가된 보안 취약점이나 위험 존재 여부 확인

소프트웨어 수명주기 동안 이루어지는 대부분의 확인 및 검증활동들은 소프트웨어의 결함을 예방하고 결함을 제거하는 것을 목적으로 한다. 소프트웨어 개발공정 동안 수행되는 결함예방태스크 중 일부는 관점에 따라 결함제거 태스크로 볼 수도 있다. 결함제거를 위해 개념단계에서는 신뢰성 확보를 위하여 시스템의 요구사항을 철저히 분석하여 소프트웨어의 개념이 잘 설정되었는지를 확인하여야 하며, 안전성확보를 위해 결함탐지, 결함고립, 결함진단, 오류복구 등에 대한 사용자의 요구사항을 확인해야 한다. 신뢰도 요구사항 및 실패에 대한 검증은 신중히 이루어져야 한다. 유사한 시스템 혹은 같은 분야의 어플리케이션으로부터 확보된 데이터가 설정된 가정을 뒷받침해주기 위해 필요할 수도 있다. 요구사항 단계에서는 신뢰성 및 안전성 확보를 위해 시스템 시험 및 승인시험 계획을 작성하여야 한다. 각종 시험의 계획 및 설계, 시험 절차 및 시험사례 작성, 시험수행은 해당 소프트웨어의 무결성수준에 맞게 이행되어야 한다 [6]. 소프트웨어 수명주기 동안의 대표적인 결함제거태스크가 표 3에 제시되어있다.

4.2 결함허용 및 결함예측 프로세스와 소프트웨어개발

결함허용프로세스는 위험도 분석을 통해 얻은 결함 및 위해도 목록들에 대해 목표하는 신뢰도를 얻을 수 있도록 기능들을 정의하고 개발하는 것을 목적으로 한다. 개념단계에서 시스템의 운영, 유지보수, 설치 및 폐기 시에 발생 가능한 결함의 탐지, 고립, 진단 그리고 오류복구 등에 대한 사용자의 요구사항을 확인해야 한다. 요구사항 단계에서는 결함 및 위험요소들에 대한 목록을 확인하고 각 위험요소에 대한 허용 가능한 수준을 정의한다. 또한 각 위험요소를 허용 가능한 수준으로 대처하기 위해 필요한 기능을 확인한다.

표 3. 결함제거 태스크와 소프트웨어 수명주기

개발단계	결함제거 태스크
개념	*신뢰성* ● 시스템 요구사항 분석 *안전성* ● 결함탐지, 결함 고립, 진단, 오류 복구 등에 대한 요구사항 확인
요구사항	*신뢰성* ● 소프트웨어 무결성수준에 따른 시스템시험 및 승인시험 계획 작성 *안전성* ● 위험도 분석 검토 및 수정 ● 위험 제거, 삭감, 완화를 위한 권고
설계	*신뢰성* ● 소프트웨어 무결성수준에 따른 컴포넌트시험, 통합시험, 시스템 시험 계획 및 설계 작성 *안전성* ● 위험 제거, 삭감, 완화를 위한 권고
구현	*신뢰성* ● 소프트웨어 무결성수준에 따른 컴포넌트시험 사례, 통합시험 사례, 승인시험 사례 및 컴포넌트시험 사례 생성 ● 소프트웨어 무결성수준에 따른 컴포넌트 시험 절차, 통합시험 절차 및 시스템시험 절차 생성 ● 소프트웨어 무결성수준에 따른 컴포넌트 시험 수행 *안전성* ● 위험 제거, 삭감, 완화를 위한 권고
테스트	*신뢰성* ● 소프트웨어 무결성수준에 따른 승인 시험 절차 생성 ● 소프트웨어 무결성수준에 따른 통합시험, 시스템 시험, 승인시험 수행

결함예측프로세스는 안전성 관점에서 매우 중요하다. 개념 및 요구사항 단계에서 신뢰도에 대한 사용자 및 시스템의 요구사항을 확인한다. 또한, 요구사항에 맞도록 신뢰도 항목들을 설정하고 그 항목들에 대해 정량적인 측정방법 및 목표 값을 설정한다. 위험도 분석을 통하여 결함 및 실패를 확인하고 심각도에 따라 분류한다. 시스템의 기능을 무결성수준에 따라 분류하고 결함을 예측하기 위한 계획을 세운다. 또한, 수집할 데이터와 분석할 계획을 잡는다.

5. 결론

원자력발전소와 같이 안전-필수 소프트웨어가 사용되는 곳에서는 소프트웨어의 신뢰도를 확보하기 위한 노력들을 계속해오고 있다. 소프트웨어의 신뢰도는 소프트웨어의 수명주기 동안 신뢰성, 안전성, 보안 등의 속성을 향상시키기 위해 수행하는 결함예방프로세스, 결함허용프로세스, 결함제거프로세스, 그리고 결함예측프로세스를 통해 가능하다. 본 논문에서는 신뢰도 향상을 위한 신뢰도 프로세스 모델과 소프트웨어의 개발공정 동안 수행되어야 하는 신뢰도 태스크들을 제시하였다. 또한, 각 개발 단계마다 수행되어야 할 신뢰도 태스크들을 신뢰도 속성에 따라 분류하였다. 본 논문에서 제시된 신뢰도 태스크들은 사용될 소프트웨어의 활용분야 및 사용자의 요구에 따라 수정되어야 할 것으로 본다.

참고문헌

[1] Laprie, J.-C., Dependability: basic concepts and terminology, dependable computing and fault-tolerant systems. Springer Verlag, Wien-New York.

[2] Paolo Donzelli, Victor Basili, A practical framework for elicting and modeling system dependability requirements: Experience from the NASA high dependability computing project, The Journal of Systems and Software, 2005

[3] A framework for dependability engineering of critical computing systems, Mohamed Kaaniche, Jean-Claude Laprie, Safety Science, Volume 40, December 2002

[4] Michael R. Lyu, Handbook of Software Reliability Engineering, 1996.

[5] US Department of Defense, 2000. Standard practice for system safety, MIL-STD-882D.

[6] IEEE Std. 1012, IEEE Standard for Software Verification and Validation, 2004