SW 신뢰성 향상을 위한 국제 규격 현황

MDS테크놀로지(주) ▌우경일 · 우준석

1. 서 론

SW 신뢰성을 “SW가 고장 또는 결함으로부터 자유로울 수 있는 정도”라고 정의한다면, 이러한 정도를 측정한다는 것이 현실적으로 쉽지 않다는 것을 쉽게 짐작할 수 있을 것이다. 이미 SW 신뢰성에 대해서는 1990년대에 미국에서 그리고 2000년대에는 유럽에서 다양한 국제 규격과 논문, 서적들이 활발하게 발간되었다. 이 후 2000년대 후반에 들어서는 연구와 적용 사례들의 발표가 다소 주춤한 상황이다.

근래 들어서는 목표 시스템을 설계, 개발, 생산하는 과정에서 단독으로 하기 보다는 생태계를 통해 다양한 형태의 협업이 이루어지는 것이 일반적이다. 더불어 시스템의 기능들이 매우 다양화 되고 지능화 되면서 그러한 시스템의 기능들을 구현하는데 SW의 역할과 비중이 매우 커지게 되어 이를 담당하는 SW는 불과 10년 전인 2000년대 초반과는 비교하지 못할 정도의 복잡도를 가지게 되었다.

이렇게 시스템의 복잡도가 높아짐에 따라 SW의 신뢰성도 매우 중요하게 되었다. 이러한 시점에서 본 기고를 통해 SW 신뢰성이 전통적으로 어떻게 다루어져 왔으며, 이와 관련된 다양한 국제 규격들에 대해 살펴보는 것은 SW 신뢰성을 향상시키기 위한 노력들이 어떤 방향으로 추구되고 있는지를 이해하는데 커다란 도움이 될 것으로 생각한다.

2. SW 신뢰성이란

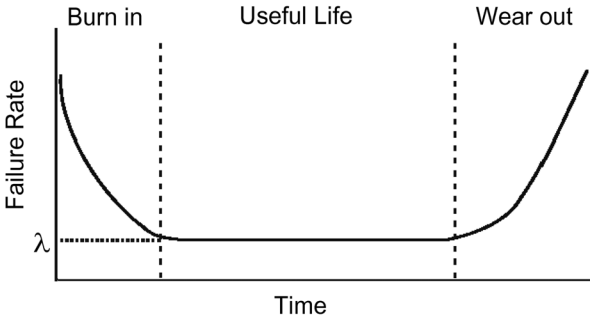
2.1 SW 신뢰성(SW Reliability)의 정의

SW 신뢰성에 대한 정의는 매우 다양하다. IEEE 610.12-1990에서는 SW 신뢰성을 “규정된 환경하에서 주어진 기간 동안 요구된 기능을 수행하는 시스템 또는 컴포넌트의 수행 능력”이라고 정의하고 있다. ANSI/IEEE STD-729-1991에서는 “특정 환경에서 일정 기간 동안 결함 없이 작동할 확률”이라고 정의하고 있으며, 이에 따라 SW 신뢰성을 0과 1 사이의 값으로 표현하기도 한다. 한편, ISO/IEC 9126에서는 SW 신뢰성을 “SW 품질특성”으로 나타내며, 성숙성(Maturity), 장애 허용성(Fault tolerance), 복구성(Recoverability), 표준적합성(Compliance)의 품질부 특성을 가진 것으로 정의하고 있다.

비교적 최근에 새롭게 정의된 개념에 의하면 SW 신뢰성은 “SW가 소기의 작동 환경에서 다양한 부하에서도 정확하고 일관된 결과들을 반복적으로 만들어 낼 수 있는가에 대한 확신할 수 있는 정도”로, SW 테스팅에 비중을 둔 해석이라고 할 수 있다(원문: Software reliability is defined as “a measure of confidence that the software produces accurate and consistent results”, which are “repeatable under low, normal and peak loads in the intended operational environment”. 출처: Software Safety and Reliability, IEEE Computer Society Press 1999).

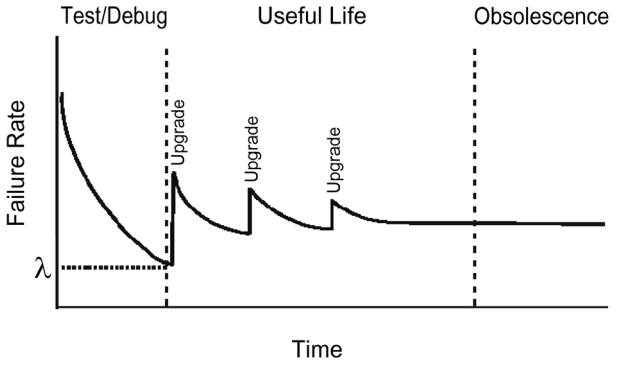
2.2 HW 신뢰성과 SW 신뢰성의 차이

신뢰성은 발생되는 고장 또는 결함이 잔존하는 확률 또는 가능성과 관계가 있다. 일반적으로 HW는 자체 특성으로 인하여 제품의 소모나 외부 요인에 의한 고장으로 인해 그 신뢰성이 저하되는 반면, SW 고장은 그 원인들이 코드 내 각 인자 값의 잘못된 설정 또는



(출처: Jiantao Pan, Software Reliability, Carnegie Mellon University, 1999)

그림 1 Bathtub Curve for HW Reliability



(출처: Jiantao Pan, Software Reliability, Carnegie Mellon University, 1999)

그림 2 Revised bathtub curve for software reliability

Logic이나 알고리즘의 잘못된 구현에서 비롯된다. HW의 경우에는 비교적 외부 환경으로부터 민감한 영향을 받으며 시간이 지나면 고장률이 급증함에 따라 신뢰도가 떨어지게 된다.

한편 SW의 경우는 양산 또는 출시 후에 활발하게 사용되면서 문제점들이 들어나게 되어 결과적으로 고장률이 증가한다. 이 후 개발자의 조치(Upgrade 또는 Patch)로 고장률이 낮추지는 단계들을 수 차례 거치는데 이러한 안정화 단계를 거치게 되면 시간의 흐름과는 상관 없이 일정한 고장률을 가지게 된다.

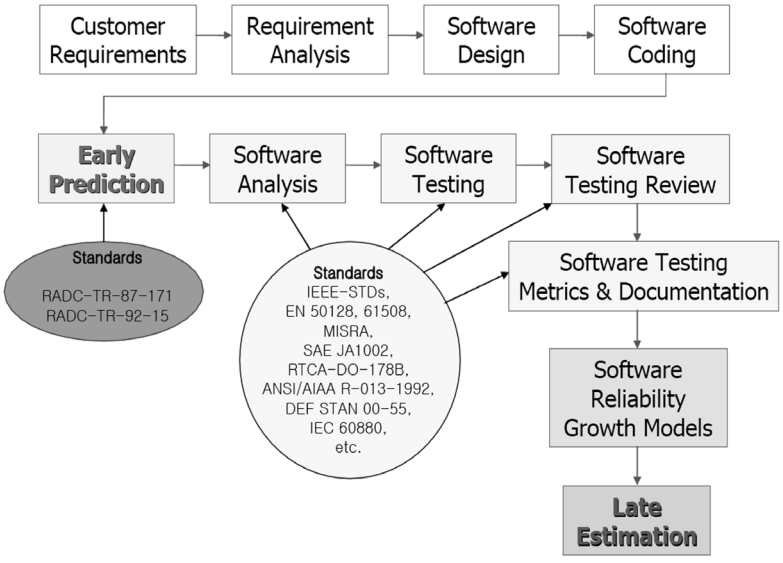
이렇게 HW 신뢰성과 SW 신뢰성은 서로 다른 양상으로 전개되므로, HW 신뢰성을 측정하는 기준을 가지고 SW 신뢰성을 측정한다면 많은 모순들이 발생하게 된다. 이에 따라 SW 신뢰성에 영향을 미치는 요소들을 정량화하기 위해 다양한 연구들이 진행되었다.

3.3 SW 신뢰성에 대한 연구 현황

ISO/IEC 9126에서 정의된 바와 같이 SW 신뢰성은 SW 품질 특성으로 나타내며 다양한 품질 부 특성을 가지고 있다고 언급하였다. 개발자의 입장에서는 사용자의 요구사항과 SW 품질이 서로 충돌하는 경우가 빈번히 일어난다. 신뢰성 목표가 너무 높게 설정되면 인도 시점이 미뤄지고, 개발 비용이 증가하게 되는 반면, 신뢰성 목표가 너무 낮게 설정되면 사용자는 큰 불만을 가지고, 산업의 특성에 따라서는 심각한 인명상의 피해 또는 재산상의 손해까지도 발생할 수 있다.

SW 신뢰성 공학은 제품이 어떻게 사용될 지와 주요 품질 요소를 정량화함으로써 이루어진다. 작동 프로파일(Operational Profile)을 작성함으로써 제품이 사용자 환경에서 어떻게 사용될지를 예측하고, 이를 토대로 자주 사용하거나 고객에게 심각한 영향을 미치는 기능을 분별해내서 개발 단계에 자원(테스트 케이스, 검토 시간 및 인원, 시험 시간)을 합리적으로 할당할 수 있도록 한다. 그리고 고객이 중요하게 여기는 품질 요소(개발 기간, 비용, 신뢰성)를 정량화함으로써 각 품질 요소의 목표치가 서로 균형을 이루게 할 수 있다.

신뢰성 예측은 신뢰성과 관련된 유용한 척도와 측도를 통해 현재 또는 미래의 신뢰성을 결정하며, 요구사항분석부터 코딩단계까지의 활동과 산출물을 통해 시험 이전단계의 신뢰성을 예측할 수 있다. 신뢰성 추정은 시험/운영 동안에 얻어진 Fault/Failure 데이터를 기반으로 현재의 신뢰성을 결정하며, 신뢰성 목표 값과 비교하여 만족할 만한 수준에서 테스팅을 중단하는 결정



(출처:The DACS Software Reliability Sourcebook, Data and Analysis Center for Software(Rome LAB), 2001.)

그림 3 SW 개발 수명 주기에서 신뢰성 예측과 추정

표 1 SW 신뢰성 예측 모델 및 추정 모델간의 차이

구분	예측 모델	추정 모델
참고수치	경험적인 수치를 사용	현재 SW 개발에 들이는 노력을 산출하여 사용
개발 프로세스 상에서 적용 시	SW 테스트 이전에 수행 또는 개념(Concept) 설정 초기 단계에서 적용	개발 프로세스 후반 부분에 적용(초기 단계에서는 적용하지 않음)
기준시점	미래 시점에서의 신뢰성을 예측	현재 또는 미래 시점에서의 신뢰성을 추정

(출처: Jiantao Pan, Software Reliability, Carnegie Mellon University, 1999)

을 할 수 있고, 신뢰성 목표 값을 달성하기 위해 추가적으로 수행되어야 할 테스팅에 필요한 시간을 예측할 수 있다.

예측모델과 추정모델은 소프트웨어 개발단계에서 사용되는 시기가 다르므로 이들 모델의 입력과 출력에도 차이가 있다. 일반적으로 예측모델을 활용하면 소프트웨어 개발 환경과 경험 그리고 개발 프로세스의 활동들로부터 수집된 데이터를 기반으로 시험 이전 단계에서의 초기 결함 밀도와 현재의 신뢰성을 예측할 수 있다. 그리고 추정모델은 시험을 통해 수집된 결함 데이터를 기반으로 해서 고장률과 남아있는 결함 수를 추정하고 목표로 하는 신뢰성을 달성하기 위해 수행되어야 할 시험 시간, 현재의 신뢰성의 추정을 가능하게 한다.

4. 국제 규격에서의 SW 신뢰성 향상 방안

이미 위에서 언급한 바와 같이 SW 신뢰성을 “특정한 환경에서 특정한 기간 동안 소프트웨어가 오류 없이 작동할 확률(ANSI/IEEE STD-729-1991)”이라고 한다면 일반적으로 소프트웨어의 신뢰성을 향상시키는 방법은 소프트웨어의 오류를 줄이는 것으로, 소프트웨어의 Verification & Validation, 시험 등의 공학적 기법을 통해 소프트웨어의 오류를 검출하고 제거하는 것이라고 할 수 있다(JiantaoPan, Software Reliability, CMU, 1999).

아래에서는 실제로 높은 신뢰도를 요구하고 있는 산업군들을 대상으로 공표되어 있는 국제 규격들을 살펴보고, 이들 규격에서는 SW 신뢰성을 향상시키기 위해 이러한 관점이 어떻게 반영되어 있는지 살펴보기로 한다.

4.1 IEC 61508(기능안전규격 SW분야)에서의 SW 신뢰성 향상 방안

기능안전규격의 모(母)규격이라고 불리는 IEC 61508-3에서는 다음과 같은 SW 테스팅들을 통해 SW 신뢰도 향상을 꾀하고 있다.

표 2 IEC 61508에서 강제 되어 있는 디자인 및 코딩 규칙과 모듈화를 구현하기 위한 수행해야 할 내용

Technique/Measure*		Ref.	SIL	SIL 2	SIL 3	SIL 4
1	Use of coding standard to reduce likelihood of errors	C.2.6.2	HR	HR	HR	HR
2	No dynamic objects	C.2.6.3	R	R	HR	HR
3a	No dynamic variables	C.2.6.3	---	R	HR	HR
3b	Online checking of the installation of dynamic variables	C.2.6.4	---	R	HR	HR
4	Limited use of interrupts	C.2.6.5	R	R	HR	HR
5	Limited use of pointers	C.2.6.6	---	R	HR	HR
6	Limited use of recursion	C.2.6.7	---	R	HR	HR
7	No unstructured control flow in programs in higher level languages	C.2.6.2	R	HR	HR	HR
8	No automatic type conversion	C.2.6.2	R	HR	HR	HR

Technique/Measure*		Ref.	SIL	SIL 2	SIL 3	SIL 4
1	Software module size limit	C.2.9	HR	HR	HR	HR
2	Software complexity control	C.5.13	R		HR	HR
3	Information hiding/encapsulation	C.2.8	R	HR	HR	HR
4	Parameter number limit/fixed number of subprogram parameters	C.2.9	R	R	R	R
5	One entry/one exit point in subroutines and functions	C.2.9	HR	HR	HR	HR
6	Fully defined interface	C.2.9	HR	HR	HR	HR

우선 SW 코딩에 대한 지침을 아래와 같이 만들어 각 등급별로 준수해야 하는 코딩 규칙들을 안내하고 있다.

위와 같은 코딩 규칙들을 준수하더라도 SW 규모가 커지면 이에 따라 각 함수 간 또는 파일 간 참조하는 것이 증가하여 복잡도가 증가하게 된다. SW 신뢰성을 향상시키기 위해서는 이러한 코딩 규칙을 통해 결함의 발생을 예방하는 활동과 함께 잠재적 결함을 검출하는 과정을 요구하고 있다. IEC 61508에서도 정적 분석을 통해 결함을 검출하는 과정을 실행하도록 요구하고 있으며, 그 내용은 다음과 같다.

이러한 활동들을 통해 코드상에서의 결함들을 검출하였다면, 코드를 실제 타겟 환경에서 작동시켜 봄으로써 설계 문서상에 명시된 동작을 수행하는지 여부를 검사한다. 이 과정을 단위시험 또는 동적시험이라고 하며, 관련 내용은 다음과 같다.

상기의 표에서도 확인할 수 있듯이, 높은 SW 신뢰성을 도달하기 위해 IEC 61508에서는 Code Coverage에 대한기준을 명확하게 규정하고 있다.

4.2 ISO 26262(자동차 SW분야)에서의 SW 신뢰성 향상 방안

거론되는 규격들 중 가장 최근에 공표(2011년 11월 15일 공표)된 자동차 분야규격인 ISO 26262는 자동차 산업의 특성(OEM ↔ Supplier에 의한 생태계를 통해 제작됨)을 반영했을 뿐 아니라, IEC 61508과는 달리 신뢰성의 기준을 양자간의 계약에 의해 결정하도록 맡겨 놓았다. 즉, 높은 신뢰도를 추구하는 것과 적기 제품 출시라는 양단 사이에서 합의점을 찾을 수 있도록 한 것이다. 하지만, 이를 자율에 맡겼다기 보다는 SW 개발 당사자들이 스스로 책임을 지도록 함으로써 상응하는 경쟁력을 보유한 업체만이 살아남을 수 있게 되었다는 해석이 더욱 적절하다는 평가를 받고 있다.

자동차 기능안전성 개발 규격인 ISO 26262에서 요구하는 SW 신뢰성 향상 방안에는 여러 가지가 포함되어 있는데 우선 다음과 같은 코딩 가이드라인을 준수하도록 하고 있다.

이러한 코딩가이드라인 뿐 아니라, 코딩을 하면서 반

표 3 IEC 61508에서 강제화 되어 있는 정적 분석 방안

Technique/Measure*		Ref.	SIL	SIL 2	SIL 3	SIL 4
1	Boundary value analysis	C.5.4	R	R	HR	HR
2	Checklists	B.2.5	R	R	R	R
3	Control flow analysis	C.5.9	R	HR	HR	HR
4	Data flow analysis	C.5.10	R	HR	HR	HR
5	Error guessing	C.5.5	R	R	R	R
6a	Formal inspections, including specific criteria	C.5.14	R	F	HR	HR
6b	Walk-through(software)	C.5.15	R	R	R	R
7	Symbolic execution	C.5.11	---	---	R	R
8	Design review	C.5.16	HR	HR	HR	HR
9	Static analysis of run tiome error behaviour	B.2.2, C.2.4	R	R	R	HR
10	Worst-case execution time analysis	C.5.20	R	R	R	R

표 4 IEC 61508에서 강제화 되어 있는 동적 분석 및 시험 항목

Technique/Measure*		Ref.	SIL	SIL 2	SIL 3	SIL 4
1	Test case execution from boundary value analysis	C.5.4	R	HR	HR	HR
2	Test case execution from error guessing	C.5.5	R	R	R	R
3	Test case execution from error seeding	C.5.6	---	R	R	R
4	Test case execution from model-based test case generation	C.5.27	R	R	HR	HR
5	Performance modeling	C.5.20	R	R	R	HR
6	Equivalence classes and input partition testing	C.5.7	R	R	R	HR
7a	Structural test coverage(entry points) 100%**	C.5.8	HR	HR	HR	HR
7b	Structural test coverage(statements) 100%**	C.5.8	R	HR	HR	HR
7c	Structural test coverage(branches) 100%**	C.5.8	R	R	HR	HR
7d	Structural test coverage(conditions, MC/DC) 100%**	C.5.8	R	R	R	HR

표 5 ISO 26262에서 요구하는 코딩 가이드 라인

Topics		ASIL			
		A	B	C	D
1a	Enforcement of low complexity	++	++	++	++
1b	Use of language subsets	++	++	++	++
1c	Enforcement of strong typing	++	++	++	++
1d	Use of defensive implementation techniques	o	+	++	++
1e	Use of established design principles	+	+	+	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++

표 6 ISO 26262에서 요구하는 코딩 규칙

Methods		ASIL			
		A	B	C	D
1a	One entry and one exit point in subprograms and functions	++	++	++	++
1b	No dynamic objects or variables, or else online test during their creation	+	++	++	++
1c	Initialization of variables	++	++	++	++
1d	No multiple use of variable names	+	++	++	++
1e	Avoid global variables or else justify their usage	+	+	++	++
1f	Limited use of pointers	O	+	+	++
1g	No implicit type conversions	+	++	++	++
1h	No hidden data flow or control flow	+	++	++	++
1i	No unconditional jumps	++	++	++	++
1j	No recursions	+	+	++	++

드시 지켜야 할 코딩 규칙도 아래와 같이 명시하고 있다.

또한 IEC 61508에서처럼 코딩 규칙을 준수함은 물론정적 분석을 통해 의미 기반의 오류 검출 과정을 거치도록 규정하고 있다.

특히 ISO 26262에서는 앞서 살펴봤던 IEC 61508보다 더욱 많은, 그리고 구체적인 동적시험을 요구하고 있으며, 그 기준이 여타 기준들 보다 자세하게 설명되고 있다. 우선, 단위시험을 통해 동적 시험을 시작하도록 되어 있으며, 단위시험에 대한 기법과 함께 단위시험에 필요한 테스트 케이스를 만들어 내는 방법도 명시를 하고 있다. 또한, 필요한 테스트 케이스를 얼마만큼 만들어야 하는지의 기준에 대해 Structural Coverage Metric을 통해 그 범위를 안내하고 있다.

4.3 IEC 62279(철도 SW분야)

높은 SW 신뢰성을 요구하고 있는 철도 분야에서도 위에서 언급했던 IEC 61508과 ISO 26262와 유사한 규정들을 포함하고 있다.

표 7 ISO 26262에서 요구하는 SW 정적 분석의 내용

Methods		ASIL			
		A	B	C	D
1a	Walk-through[a]	++	+	O	O
1b	Inspection[a]	+	++	++	++
1c	Semi-formal verification	+	+	++	++
1d	Formal verification	O	O	+	+
1e	Control flow analysis[b,c]	+	+	++	++
1f	Data flow analysis[b,c]	+	+	++	++
1g	Static code analysis	+	++	++	++
1h	Semantic code analysis[d]	+	+	+	+

표 8 ISO 26262에서 요구하는 동적 시험의 내용/방법/기준들

Methods		ASIL			
		A	B	C	D
1a	Walk-through[a]	++	+	O	O
1b	Inspection[a]	+	++	++	++
1c	Semi-formal verification	+	+	++	++
1d	Formal verification	O	O	+	+
1e	Control flow analysis[b,c]	+	+	++	++
1f	Data flow analysis[b,c]	+	+	++	++
1g	Static code analysis	+	++	++	++
1h	Semantic code analysis[d]	+	+	+	+

Methods		ASIL			
		A	B	C	D
1a	Requirements-based test[a]	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test[b]	+	+	+	++
1d	Resource usage test[c]	+	+	+	++
1e	Back-to-back comparison test between model and code, if applicable[d]	+	+	++	++

Methods		ASIL			
		A	B	C	D
1a	Analysis of requirements	++	++	++	++
1b	Generation and analysis of equivalence classes	+	++	++	++
1c	Analysis of boundary values	+	++	++	++
1d	Error guessing	+	+	+	+

Methods		ASIL			
		A	B	C	D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++

표 9 IEC 62279에서 요구하는 코딩 규칙/오류 검출/동적 시험의 내용, 방법, 기준 항목

TECHNIQUE/MEASURE	Ref	SW SIL0	SW SIL1	SW SIL2	SW SIL3	SW SIL4
1. Coding Standard Exists	B. 16	HR	HR	HR	HR	HR
2. Coding Style Guide	B. 16	HR	HR	HR	HR	HR
3. No Dynamic Objects	B. 16	–	R	R	HR	HR
4. No Dynamic Variables	B. 16	–	R	R	HR	HR
5. Limited Use of Pointers	B. 16	–	R	R	R	R
6. Limited Use of Recursion	B. 16	–	R	R	HR	HR
7. No Unconditional Jumps	B. 16	–	HR	HR	HR	HR

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Boundary Value Analysis	B. 4	–	R	R	HR	HR
2. Checklists	B. 8	–	R	R	R	R
3. Control Flow Analysis	B. 9	–	HR	HR	HR	HR
4. Data Flow Analysis	B. 11	–	HR	HR	HR	HR

5. Error Guessing	B. 21	–	R	R	R	R
6. Fagan Inspections	B. 24	–	R	R	HR	HR
7. Sneak Circuit Analysis	B. 55	–	–	–	R	R
8. Symbolic Execution	B. 63	–	R	R	HR	HR
9. Walkthroughs/Design Reviews	B. 66	HR	HR	HR	HR	HR

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Test Case Execution from Cause Consequence Diagrams	B. 6	–	–	–	R	R
2. Prototyping/Animation	B. 49	–	–	–	R	R
3. Boundary Value Analysis	B. 4	R	HR	HR	HR	HR
4. Equivalence Classes and Input Partition Testing	B. 19	R	HR	HR	HR	HR
5. Process Simulation	B. 48	R	R	R	R	R

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Test Case Execution from Boundary Value Analysis	B. 4	–	HR	HR	HR	HR
2. Test Case Execution from Error Guessing	B. 21	R	R	R	R	R
3. Test Case Execution from Error Seeding	B. 22	–	R	R	R	R
4. Performance Modeling	B. 45	–	R	R	HR	HR
5. Equivalence Classes and Input Partition Testing	B. 19	–	R	R	HR	HR
6. Structure-Based Testing	B. 58	–	R	R	HR	HR

5. SW 신뢰성을 향상 시키는 방안

위에서 살펴본 바와 같이 대표적으로 고 신뢰성을 요구하는 산업에서 SW 신뢰성을 향상시키기 위해 요구하는 항목들에는 공통적 요소가 많이 있다는 것을 확인할 수 있었다. 이번에는 관련 내용들을 좀 더 구체적으로 살펴보기로 한다.

5.1 Coding Rule 제정을 통한 관리

위에서 언급했듯 SW 신뢰성을 저하시키는 요인들은 외부 환경적인 요인들이 아닌 개발자들이 코딩하는 과정에서 대다수 실수로 만들어 내는 것이 많다. 뿐만 아니라, ISO C90/C99 조차도 모호한 표현들이 들어 있어, ISO의 관련 표준에 대한 교육을 수료하지 않는 대다수 개발자들은 자신도 모르는 사이에 잘못된 표현을 사용하게 된다. 이러한 실수와 모호함들을 해결하기 위해 SW 코딩 룰을 제정하여 코딩 초기 단계에서부터 잠재적 결함들을 예방하는 것이 바람직하며, 이를 위해 QAC/ QAC++와 같은 도구들의 도움을 받는 것이 일반적이다.

5.2 잠재적 오류 검출 활동

실제 코딩 규칙을 제정하여 개발자들이 준수하게 함으로써 많은 실수들을 예방하였다고 하더라도, 다수의 개발자가 협업을 통해 개발하는 프로젝트의 경우 각 개발자들의 산출물들을 통합하는 과정에서 많은 오류

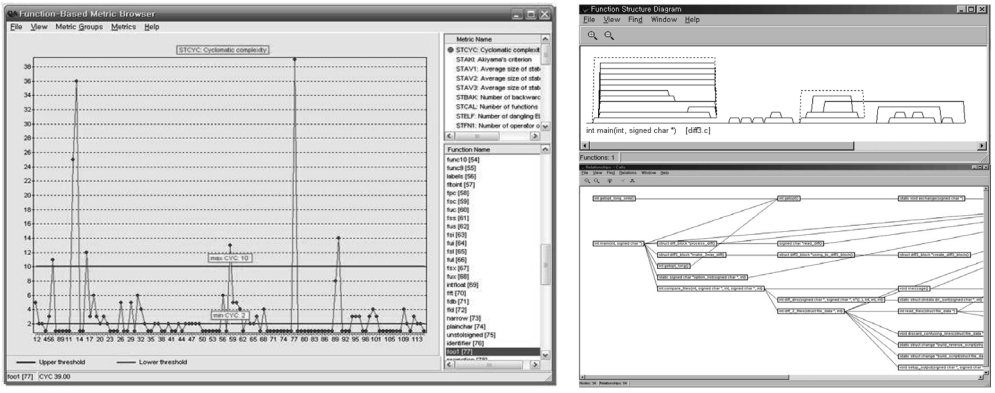


그림 4 QAC가 코딩 룰을 기반으로 한 SW Metrics와 함수/Call 관계를 보여주는 화면

들이 발생되고 있다. 국제 규격에서도 코딩 규칙을 제정하여 적용함과 동시에 정적분석 활동을 수행하도록 안내하고 있으며, 이를 통해 잠재적인 결함들을 최대한 검출해 내는 것이 바람직하다. 이는 특히 결함을 일찍 발견하여 제거할수록 작은 비용이 소요된다는 사실을 감안하면 매우 효과적이다. 이러한 기술을 지원하는 도구들은 이미 산업계에 일반화 되어 사용되고 있으며 자동차나 국방 분야의 개발자들이 많이 사용하고 있는 CodeSonar와 같은 도구가 이에 해당한다.

5.3 S/W Unit Testing 수행

이렇게 코드를 수행하지 않고 잠재적 결함 여부들을 확인한 후에도, 실제의 타겟 환경에서 정확하게 수행을 하는지 설계 단계에서 도출된 테스트 케이스를 기반으로 확인하는 테스팅 과정이 필요하다. 이는 동적시험 중 단위/통합 시험에 해당 되며, 본 과정을 수행하기 위해서는 반드시 설계단계에서의 테스트 케이스들이 요구된다.

단위시험이란, 내가 원하는 하나의 소프트웨어 모듈이 정상적으로 기능을 수행하는지 여부를 시험하는 최소 수준의 시험을 말한다. 일반적으로 원시 코드를 대상으로 하며, 단위 시험을 수행하는데 사용하는 주된 시험 방법은 화이트 박스 시험(White Box Test)이다. 단위 프로그램 별로 설계서 상에 정의된 기능을 제대로 수행하는지 검증하는 것을 목적으로 한다. 단위 시험은 해당 개발자도 수행할 수 있으나 다른 개발자 또는 제3자에 의한 시험이 권장된다.

5.4 Code Coverage 확보

동적시험 중 가장 객관적으로 측정이 가능한 기법 중 하나가 Code Coverage이다. 대다수 국제규격들은 SW 신뢰성을 측정함에 있어 Code Coverage를 기준으로 삼고 있으며, 고 신뢰성을 요구하는 항목일수록 더욱 엄밀한 Code Coverage 수준을 요구하고 있다.

Code Coverage의 경우 단위/통합시험과 함께 동적시험으로 분류가 되며, 테스트 케이스를 수행함에 따라 그 결과와 함께 Code Coverage를 함께 측정하는 것이 일반적이다. 단, 동적시험은 실제 Target 환경 또는 가능

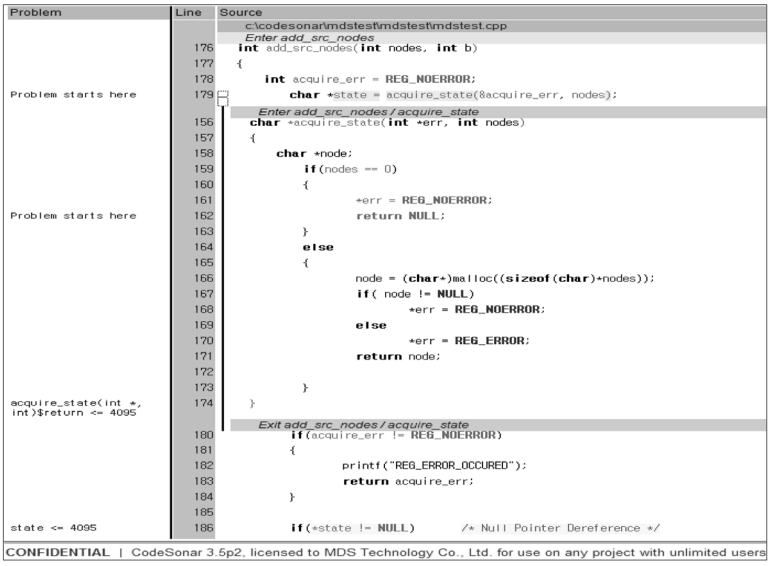


그림 5 CodeSonar를 통해 Data/Control Flow 기법으로 Null 값 참조에 따른 오류를 검출한 화면

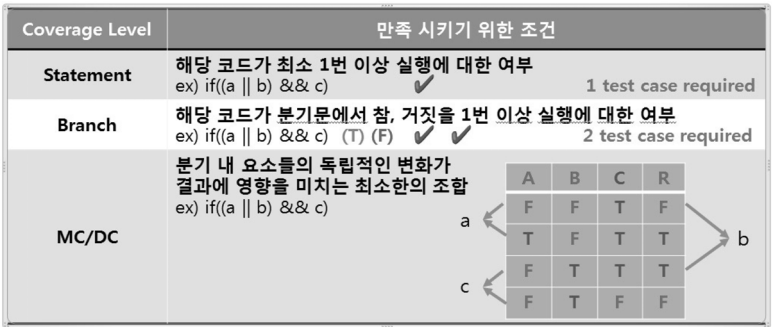


그림 6 Code Coverage 종류에 대한 설명

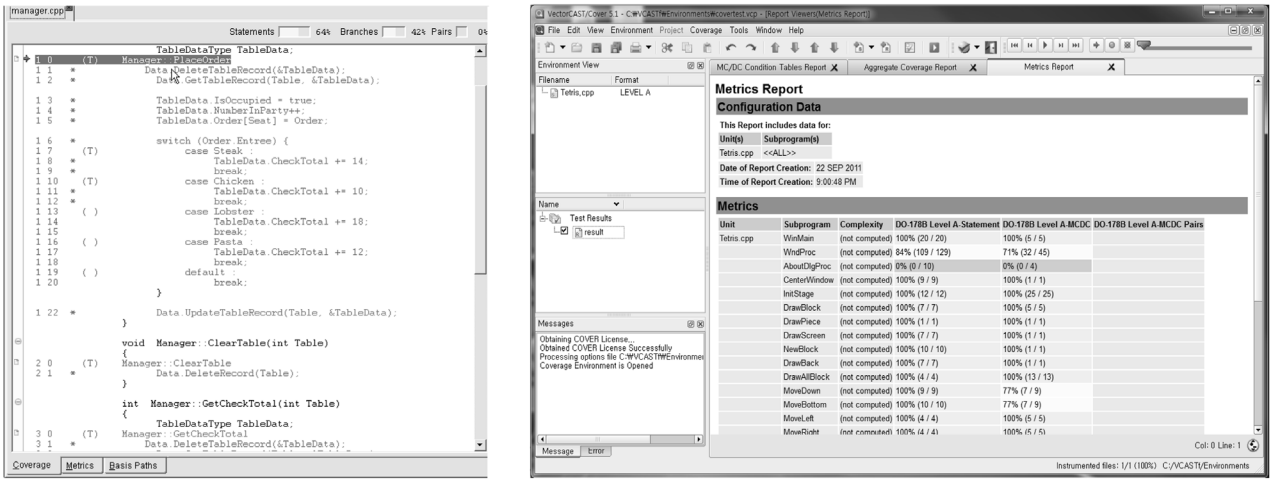


그림 7 VectorCAST를 통해 실제 Code Coverage가 측정된 Code의 화면과 요약 Report

한 이와 유사한 환경(Emulator, Simulator)에서 수행을 해야 하며, 실제 Target 환경이 아닐 경우에는 그 모사된 환경에서 생성된 환경이 실제 환경과 어떠한 차이가 발생하는지 시험 결과서에 명시를 하고 전문그룹으로부터 검토를 받도록 되어 있다. 아래의 화면은 실제 Target 환경에서 동적시험 관련 3자 검증을 수행하면서 확보한 Code Coverage 화면과 고객사에 제공되는 VectorCAST의 Report 화면이다. 근래에는 자동차 시장과 국방 시장에서 SW 신뢰성을 향상시키기 위해 동적시험을 강도 높게 강화하고 있으며, 그에 따라 전 세계적으로 성능이 검증된 도구로 다양한 활동들이 수행되고 있다.

6. 결 론

SW 신뢰도를 측정하고 이를 높이기 위한 다양한 연구들은 앞으로도 계속 진행될 것이다. 개발 비용과 시간에 대한 압박 속에서도 SW에 대한 신뢰성 요구는 반대로 더욱 커지고 있는 상황이다. 이러한 상황에 대처하기 위해 SW 신뢰성 공학이 적극적으로 활용되고 있으며 SW 신뢰성에 대한 다양한 예측과 추정 모델들이 개발되고 있다. 각 산업별로 제정되어 있는 국제 규격에서는 SW 신뢰성이 오류없이 동작할 확률이라는 정의를 SW 테스팅 관점에서 해석하여 코딩 규칙 적용, 잠재적 오류 검출, 동적 시험(단위시험, Code Coverage 측정) 등을 통해 해당 제품/프로젝트에 대한 SW 신뢰성을 평가할 뿐만 아니라 System/HW 개발에 대한 가이드라인도 함께 제시하고 있다. 특히 3~5년간의 장기 개발 기간이 소요되며 높은 신뢰성을 요구되는 자동차 관련 규격에서는 양산에 대한 경제적인 비용까지 고려해야 함에도 불구하고 신뢰성을 높이기 위해 필요한 사항들을 폭 넓게 반영하고 있으며 더욱 엄격히 관리하도록 요구하고 있다.

SW 신뢰성에 대한 관심도는 점차 높아지고 있으며 SW 신뢰성을 향상시키기 위한 활동들은 점점 그 중요성을 인정받고 있다. 이제는 SW 신뢰성 부서의 담당자들뿐만 아니라, 개발에 참여하는 모든 사람들이 SW 신뢰성을 높이기 위해 적극 노력해야 한다. 특히 각 산업별로 제정되어 있는 관련 국제 규격들에 관심을 가지고 이를 내재화 하기위한 전문성 있는 신뢰성 향상 활동이 매우 필요한 시점이다.

참고문헌

[1] Debra S. Herrmann: Software Safety and Reliability, IEEE Computer Society Press, 1999, ISBN 0-7695-0299-7, Pages 21, 22, 25, 2

[2] Ian Sommerville: Software Engineering 8, Addison-Wesley, 2007, ISBN 0-321-31379-8, Several Page

[3] Daniel Sonnick: Reliable Date-Replication Using Grid Computing Tools, 04. Mai 2009, Pages 3,4

[4] Jiantao Pan: Software Reliability, Carnegie Mellon University, 1999

[5] Jiantao Pan: Software Testing, Carnegie Mellon University, 1999

[6] John D. Musa:Introduction to Software Reliability Engineering and Testing, 8th International Symposium on Software Reliability Engineering(Case Studies). November 2-5, 1997. Albuquerque, New Mexico

[7] John D. Musa: Anthony Iannino, and Kazuhira Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill Book Company, 1987, ISBN 0-07-044093-X

[8] Musa, J: “Operational Profiles in Software-Reliability Engineering”, IEEE Software, March 1993
[9] John D. Musa: More Reliable Software Faster and Cheaper, Software Reliability Engineering and Testing Courses
[10] Taewan Gu: Software Reliability Engineering, (taewan.blogspot.com/p/software-engineering.html)
[11] IEC 62279 : Railway applications-Communications, signalling and processing systems-Software for railway control and protection systems
[12] ISO 26262-6 : Road vehicles-Functional safety-Part 6: Product development at the software level
[13] IEC 61508-3 Ed. 2.0: Functional safety of electrical/electronic/programmable electronic safety related systems-Part 3: Software requirements
[14] Functional Safety, A Straightforward Guide to applying IEC 61508 and Related Standards, Second edition, David J Smith & Kenneth G L Simpson, ISBN 0-7506-6269-7

약 력

우 경 일

1997~2002 유한대학 정보통신공학과 졸
2005~2007 한국방송통신대학교 경영학과 학사
2001~2003 오성테크
2003~2005 대신네트웍스
2005~현재 MDS테크놀로지 TC사업부 TA사업팀장
관심분야 : SW 품질, SW 신뢰성, SW 테스팅, ISO 26262, 모바일/국방/자동차/철도 SW 산업 등
E-mail : kyungil@mdstec.com

우 준 석

1985~1990 서울대학교 전자공학과 졸
1990~1992 한국과학기술원 전기공학과 석사
1992~1999 LG전자
2000~2002 새롬기술
2003~현재 MDS테크놀로지 TC사업부장/상무
관심분야 : 모델 기반 설계, 시험 자동화, 시스템 엔지니어링, HILS 시뮬레이션, AUTOSAR, ISO26262
E-mail : Joonseok@mdstec.com