의료기기 소프트웨어 정적검증을 통한 임베디드 코딩룰과 실행시간 오류 간 연관성 분석

유새열, 최기용, 이정원
아주대학교 전자공학과
e-mail:dbtoduf@ajou.ac.kr, ki815kaisian@ajou.ac.kr, jungwony@ajou.ac.kr

An Analysis of the Relation between Runtime Errors and Embedded Coding Rule using Static Analysis of Medical Software

Sae-Yeol Yoo, Ki-Yong Choi, Jung-Won Lee
Dept. of Electrical and Computer Engineering, Ajou University

요 약

최근 임베디드 소프트웨어의 신뢰성과 안전성을 보장하기 위하여 코딩룰인 MISRA-C를 자동차 뿐만 아니라 군사, 의료 분야등 광범위한 분야에서 이용하고 있다. 하지만 MISRA-C가 자동차 시스템분야의 MISRA 가이드를 이용하여 개발되었기 때문에 타 분야의 분야별 특성을 모두 고려하지 못한다는 문제점이 제시되고 있다. 따라서 본 논문에서는 향후 의료기기 분야의 특성을 고려한 코딩룰을 제시하기에 앞서, 의료기기 소프트웨어에서의 코딩룰 필요성을 제시한다. 이를 위해 개발 단계의 의료기기 소스코드에 MISRA-C를 적용하여 정적 분석을 해보고, 적용 유무 따른 실행시간 오류 결과를 분석한다. 분석 결과, 코딩룰을 이용하면 실질적으로 실행시간 오류 발생을 막을 수 있고, 적용 과정에서 기타 다른 실행시간 오류들 또한 해결됨을 확인하였다. 위 결과로 본 논문에서는 의료 분야의 특성을 고려한 특화 코딩룰의 필요성을 제시한다.

1. 서론

최근 의료기기가 점차 복잡화되고 신기술이 도입되면서 기기제어를 목적으로 하는 의료기기용 소프트웨어의 중요성이 빠르게 증가하고 있다[1]. 예를 들어 최근의 약물 자동 주입기는 170,000의 LOC(Line Of Code)를 가지고 있고 복잡한 의료용 소프트웨어의 경우에는 수 천 LOC부터 수백만 LOC를 가지고 있다[2,3]. 이와 같은 추세에 따라 소프트웨어 결함으로부터 오는 문제점 역시 증가하고 있다. 미 FDA(Food and Drug Administration)는 소프트웨어 결함으로 인한 리콜이 1996년에는 10%였지만 2006년에는 21%로 증가했고, 앞으로 매년 2~3배씩 증가할 것으로 예측하고 있다[1,3].

의료기기 소프트웨어의 경우에는 결함이 환자의 생명과 직결되기 때문에 안전성을 우선적으로 고려해야한다[1]. 2005년부터 2009년까지 미 FDA로 보고된 710건의 사망사고는 약물 자동 주입기 문제와 관련이 있었고, 이 중, 많은 수의 사고가 의료기기 소프트웨어의 오동작과 관련이 있었다[3]. 이러한 문제를 해결하기 위해서 의료기기 분야에서는 개발 단계별로 기법과 지표를 제시하고 국제표준을 제정하여 안전성을 확보하고 있다[1]. 그러나 V&V(Validation&Verification)단계나 개발 절차에서의 연구에 비해 코딩룰 자체에 관한 연구는 미비하다.

반면 안전성을 중요시하는 차량 분야의 경우엔 일찍이 안전한 코딩을 위해 MISRA-C(Motor Industry Software Reliability Association-C)라는 코딩룰을 정립하였다. 현재 MISRA-C는 자동차 분야 외에도 안전성이 중요한 철도, 군사, 의료와 같은 많은 임베디드 시스템 분야에 활용되고 있다[4,5]. 하지만 MISRA-C로는 각 분야 별 특성에 따른 요구사항을 모두 고려할 수 없다는 문제점이 존재한다[4,5]. 따라서 무기체계 분야나 전투기 분야의 경우에는 MISRA-C를 기반으로 분야 특성을 고려한 자체 코딩룰을 정립하여 사용하고 있다. 하지만 아직까지 의료기기 분야의 경우에는 이러한 연구가 부족하고 분야의 특성에 맞춘 정형화된 코딩룰이 없는 실정이다[4,5].

따라서 본 논문에서는 개발 단계의 의료기기 소스코드에 MISRA-C를 적용해보고, 적용 유무에 따른 실행시간 오류 결과 분석을 통해 의료기기 분야에 특화된 코딩룰 필요성을 제시하였다. 또한 이를 통해 향후 연구에서는 MISRA-C에 의료기기 분야의 특성을 도입하여 규칙들을 분류하고 추가, 제거한 의료기기용 소프트웨어 특화 코딩룰을 생성하려고 한다.

"본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터 육성지원사업의 결과로 수행되었음" "IITP-2016-R2718-16-0015"

2. 관련 연구

2.1 의료기기 SW 안전성 기술 및 표준화 동향

의료기기 소프트웨어에서의 안전성 확보가 중요해짐에 따라 이를 위한 안전성 기술 및 표준화에 대한 연구가 많이 존재한다. 하드웨어에서 안전성을 관리하기 위해 사용하는 고장발생률(Failure Rate) 개념은 소프트웨어에 적용하기가 힘들기 때문에, 소프트웨어의 경우에는 개발 단계별로 기법과 지표를 제시하여 안전성 확보를 한다[1]. 따라서 이렇게 제정된 국제표준을 소프트웨어 개발 단계에 적용하고 있다. 또한 소프트웨어 기능 안전성을 향상시키기 위해 위험관리, 품질관리, 소프트웨어 엔지니어링을 도입하고 소프트웨어 Validation활동을 통해 검증하고 있다.

하지만 대부분의 의료기기 안전성 관련 연구들은 개발 과정이나, 검증단계를 대상으로 하고 있기 때문에, 코드 작성 단계에서의 코딩룰과 같은 연구는 자동차 분야에 비해 미비한 상황이다.

2.2 정적 분석도구

소프트웨어의 신뢰성을 검사하고 오류를 찾는 방법에는 크게 동적 분석(Dynamic analysis)과 정적 분석(static analysis)방법으로 분류할 수 있다. 동적 분석은 실제 프로그램을 테스트 데이터에 의해 실행하는 방식이고, 정적 분석은 프로그램 코드를 실행하지 않고 기호 실행 기법(symbolic execution techniques)을 사용하여 코드 검토를 하여 오류를 찾는 방법이다[2,4]. 정적 분석의 경우에는 동적 테스트 전에 결함을 발견할 수 있기 때문에 수정 비용 측면에서 상당히 유리하다[6].

정적 분석을 위해 사용 가능한 도구로는 CodeCheck, LDRA TestBed, CodeSonar등이 있다[4]. 본 논문에서는 의료기기 소스코드의 MISRA-C:2004 적용 유무에 따른 안전성 변화를 확인하기 위해 정적 분석 도구 중 하나인 CodeSonar를 사용하여 개발 단계의 의료기기 소스코드에 MISRA-C:2004를 적용한 후, 정적 분석하고 적용하지 않았을 경우와 비교하였다.

2.3 CodeSonar를 사용한 의료기기 SW 분석

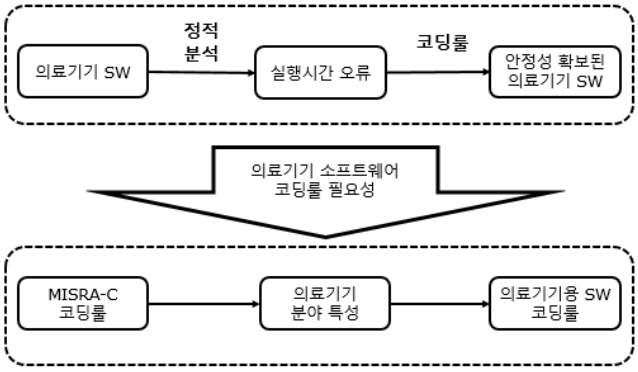
CodeSonar는 코드의 컴파일 과정을 분석하여 모든 실행 경로를 파악하고, 실행시간 오류를 검출한다. 먼저, 미FDA의 직원들이 CodeSonar를 이용하여 의료기기 소스코드를 분석했고, 어떻게 오류 검출을 하는지 설명해 놓았다. 또한 정적 분석한 결과와 제조사가 발견한 오류 분석 결과와 비교해봤다[2]. 그 결과, CodeSonar를 이용했을 때, 제조사에서 발견한 118개의 오류 중 82개가 발견되었고, 제조사에서 발견하지 못한 45개의 결함이 추가적으로 발견되었다. 또한, CodeSonar를 통해 가장 많이 검출되는 오류 중 가장 빈번히 발생하는 상위 5개의 유형의 오류만 제거하더라도 전체 결함의 평균 70%이상을 줄일 수 있다는 연구가 존재한다[6].

하지만 이러한 연구들은 모두 코드 작성을 끝낸 상태에서, 검증 단계에서 코드 검증만을 대상으로 하고 있기 때문에 코드 작성 단계에서의 코딩룰 적용 유무에 따른 오류와의 관계에 대해서 연관 지을 수 없다는 한계가 있었다.

3. 의료기기 소프트웨어 코딩룰

3.1 연구 방법

그림 1은 코딩룰을 통한 안전성 확보를 위한 연구의 프로세스를 나타낸 것이다. 각 과정은 아래와 같다.



(그림 1) 코딩룰을 통한 의료기기 SW 안전성 확보

- 정적 분석: 코딩룰이 적용되지 않은 의료기기용 SW를 정적 분석 도구를 통해 분석하여 실행시간 오류를 검출.
- 코딩룰: 검출된 실행시간 오류 해결을 위해, MISRA-C의 Rule을 적용하여 의료기기SW의 소스코드를 수정하고 안전성이 확보된 의료기기SW 생성.
- 필요성 제시: 두 SW의 실행시간 오류를 분석하고, 그 결과로부터 코딩룰 적용의 필요성을 확보.
- 의료기기용 SW 코딩룰 생성: 앞서 제기된 필요성에 근거하여, MISRA-C로부터 의료기기 분야 특성을 적용한 의료기기 특화 코딩룰 제시.

본 논문에서는 위와 같은 프로세스에 근거하여, 안정성이 확보된 SW를 얻음으로써, 의료기기 분야의 코딩룰 적용의 필요성을 제시한다.

3.2 실행시간 오류: CWE

실행시간 오류(Runtime Error)는 프로그램이 실행되는 중에 발생하여 실행에 영향을 미치는 오류이다. 코드 문법상으로는 문제없이 컴파일이 되기 때문에 개발단계에서 놓치기 쉬운 오류이다. 만약 오류가 발견되지 못한 채로 환자들이 사용한다면 사용 중에 예상치 못한 결함이 발생할 수 있다. 이러한 예상치 못한 결함은 환자의 생명까지 영향을 줄 수 있기에 더욱 신경써야하는 부분이다. 정적 분석 도구를 이용하면 이러한 실행시간 오류를 찾아낼 수 있다.

실행시간 오류를 검출할 때 사용되는 것이 MITRE에

서 만든 CWE(Common Weakness numeration)이다. CWE는 SW 취약성 관리에 대한 이해를 높이고, SW의 보안 및 품질 강화를 위해, SW 취약성을 표준화해 놓은 것이다. 그 중 CWE-658은 C 언어에서 개발자가 범하기 쉬운 공통적인 코드 결함 목록을 담고 있다. 정적 분석 도구는 이 기준에 따라서 정의된 취약성을 감지하고, 실행시간 오류를 검출한다[7].

따라서 본 논문에서는 CodeSonar를 이용해, 개발 단계 의료기기 소스코드의 실행시간 오류를 검출 및 분석한다.

3.3 코딩룰: MISRA-C

프로그램 개발 중, 역할 분담 또는 견고하지 못한 소프트웨어로부터 오는 문제를 막기 위해, 문법이 요구하는 것 이상으로 코딩 방식을 규제하는 것이 코딩룰이다. 특히 코딩룰 준수는 프로그래밍 오류를 사전에 방지할 수 있다는 점 때문에, 높은 신뢰성과 고품질이 요구되는 소프트웨어에서 매우 중요하다. 의료기기 분야의 경우는 기기의 결함이 환자의 생명에 치명적이기 때문에, 이러한 코딩룰은 더욱 중요할 수 있다.

프로그램의 목적에 맞게 CERT, MISRA-C 등 여러 가지의 코딩룰이 존재한다. 그 중 MISRA-C는 차량용 소프트웨어 신뢰성 향상을 위한 코딩룰이다. 많은 신뢰성, 고품질이 요구되는 여러 분야에서 아직까지 분야 특성을 고려한 정형화된 코딩룰이 없기 때문에 MISRA-C가 사용되고 있다. 그러나 MISRA-C가 각기 다른, 분야별 특성을 모두 만족할 수 없기 때문에, 일부 분야에서는 MISRA-C를 기반으로 하여 분야 특성을 고려한 자체 코딩룰을 사용하고 있다. 하지만, 더욱 신뢰성이 중요하게 여기지는 의료기기 분야에는 아직까지 분야 특성을 고려한 코딩룰이 없는 상황이다.

본 논문에서는 정적 분석 후, 나온 결과에 가장 최신의 MISRA-C:2012가 아닌 현재 가장 널리 사용되고 있는 MISRA-C:2004를 사용하여 실행시간 오류를 해결한다. 이 단계를 거쳐 신뢰성이 확보된 소스코드를 얻고, 이로부터 의료기기 분야의 특화 코딩룰의 필요성을 제시한다.

4. 실험 및 결과

4.1 실험 계획 및 방법

개발 단계의 의료기기 소스코드에 정적 분석 도구인 CodeSonar를 이용해서 실행시간 오류를 검출했다. 그 후, 코딩룰 적용 유무에 따른 실행시간 오류 비교를 위해 MISRA-C:2004를 같은 소스코드에 적용하였다. 그 후, 코딩룰이 적용된 개발 단계의 의료기기 소스코드에 다시 한 번 CodeSonar를 이용해, 실행시간 오류를 검출하였다. 그리고 이렇게 나온 두 결과 사이의 차이를 분석하여 실제로 MISRA-C:2004를 적용했을 때 실행시간 오류가 얼마나 개선되는지 확인해보았다.

4.2 실험 결과

먼저 실행시간 오류를 검출하기 위해, 개발 단계의 의료기기 소스코드를 CodeSonar로 정적 분석하여 표 1의 결과를 얻었다.

<표 1> CodeSonar를 통한 실행시간 오류 검출 결과

Class		Procedure	Line	Significance
(1)	Unreachable Computation	usart0_rx_isr	185	Reliability
(2)	Redundant Condition		183	Redundancy
(3)	Unused Value		182	Redundancy
(4)	Unused Value	getchar	204	Redundancy
(5)	Empty while Statement		203	Redundancy
(6)	Missing Return Statement		201	Redundancy

검출된 오류에 코딩룰을 적용해서 해결하기 위해 검출된 실행시간 오류와 MISRA-C:2004를 Mapping하고 해당 Rule을 담은 아래와 같은 표 2를 얻었다. 이때, MISRA-C와 오류는 1대1 Mapping이 되지 않기 때문에, 두 가지 오류에 대해서는 해당하는 Rule이 없다.

<표 2> 실행시간 오류/MISRA-C:2004 Mapping table

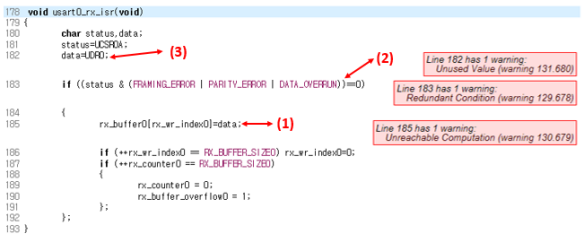
Class	MISRA-C :2004	Rule
(1)	Rule 14.1	There shall be no unreachable code.
(2)	Rule 13.7	Boolean operations whose results are invariant shall net be permitted.
(3) (4)	Rule 14.2	All non-null statements shall either: (a) have at least one side-effect however executed, or (b) cause control flow to change.
(5)	-	-
(6)	-	-

표 1, 2를 통하여, 개발 단계의 의료기기 소스코드에 두 번에 나눠 MISRA-C:2004를 적용하였고 그 결과, 실행시간 오류가 모두 제거된 안전성이 확보된 소스코드를 확보할 수 있었다.

4.3 실험 분석

표 1을 보면 검출된 오류가 Usart0_rx_isr과 getchar 두 함수에 존재한다. 따라서 각 함수 별로 표 2을 참고하여 오류를 분석하고 Rule을 적용했다.

그림 2의 (1), (2), (3) 3가지 실행시간 오류가 발생하는 부분이 있다. 오류를 분석한 결과, (2)로 검출된 if문의 조건 부분에 값이 변하지 않는 문제가 있었다. Rule 13.7에 따르면, 조건 부분의 값이 상황에 따라 변하지 않고 고정된 Boolean 데이터의 결과가 나오면 안 되었는데 실행 경로를 따라서 분석해보면 그 값이 항상 false로 일정했다. 따라서 if문 내부가 어떠한 경로를 통해서도 실행이 되지 않아서 (1)을 발생시켰고, 그 내부에서 data 변수가 사용되지 못하여 (3)을 발생시킨 것이었다. 그러므로 (2)의 오류에 Rule 13.7을 적용하여 문제를 해결하였다.



(그림 2) Usart0_rx_isr 함수 내의 실행시간 오류

그 과정에서 표 3과 같이 상수로 레지스터 주소 값을 할당한 UCSR0A나 UCR0 부분을 물리주소에 포인터를 이용해서 주소 값을 할당하는 방식으로 바꿨다. 그러자 if문 내부의 조건을 항상 false로 인식하지 않았고 true의 경로가 생기게 되었다.

<표 3> 포인터를 이용한 레지스터 주소 할당

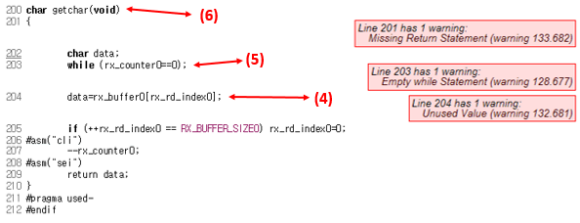
수정 전	수정 후
sfrb UCSR0A=0xb; sfrb UDR0=0xc;	#define UCSR0A (*(unsigned char *) 0x2B) #define UDR0 (*(unsigned char *) 0x2C)

그 결과, 표 4에서처럼 두 번째 분석에서 관련된 실행시간 오류 3개가 모두 해결된 것을 확인했다. 실제로는, Rule 13.7을 적용하여 (2)만 해결했지만, 이로 인해 발생했던 (1)과 (3) 모두 해결 되었다.

<표 4> 코드에 코딩룰을 적용한 결과

Class	Procedure	Line	MISRA-C: 2004	분석1	분석2	분석3
(1)	usart0_rx_isr	185	Rule 14.1	o		
(2)		183	Rule 13.7	o		
(3)		182	Rule 14.2	o		
(4)	getchar	204	Rule 14.2	o	o	
(5)		203	-	o	o	
(6)		201	-	o	o	

첫 번째 정적 분석 후, 그림 3의 getchar 함수 내 3개의 실행시간 오류 중, (4)만 MISRA-C와 Mapping이 되어 있었다. 따라서 (4) 오류에만 14.2 Rule을 적용하기 위해 원인을 분석했다. 그 결과, (4)가 발생한 이유가 (5)의 비어있는 while문 때문이라는 것을 확인했고, while문 내부를 그림 4처럼 if문 구조로 바꿔 해결했다.



(그림 3) getchar 함수 내의 실행시간 오류

```
while(1)
{
    if(rx_counter0==0);
    else
    {
        data=rx_buffer0[rx_rd_index0];
        if (++rx_rd_index0 == RX_BUFFER_SIZE0) rx_rd_index0=0;
#asm("cli")
        --rx_counter0;
#asm("sei")
        return data;
    }
}
```

(그림 4) if문을 이용해서 수정된 소스코드

그 결과, 표 4의 분석 3에서처럼 실행시간 오류가 모두 사라졌다. 코딩룰로 제거할 수 있는 오류 한 가지만 제거했음에도, 다른 부분에 영향을 끼쳐 모든 실행시간 오류가 제거된 것을 볼 수 있다. 즉, 위험할 수 있는 소스코드를 코딩룰에 따라 수정함으로써 잠재된 위험인 실행시간 오류가 해결된 것이다.

5. 결론 및 향후 계획

본 논문에서는 의료기기 분야의 특성을 고려한 의료기기 코딩룰을 생성하기에 앞서, 의료기기 분야에 임베디드 코딩룰을 적용해보고 결과를 분석해보았다. 그리고 분석결과를 통해, 의료기기 분야 코딩룰의 필요함을 알 수 있었다.

코딩룰을 적용함으로써, 실행시간 오류가 제거 되고, 다른 실행시간 오류까지 제거된 것을 확인할 수 있었다. 이를 통해서 의료기기 분야에도 SW의 안전성 확보를 위한 코딩룰이 필요하다는 것을 알 수 있다.

향후 연구에서는 의료기기 분야의 특성인 위험도와 발생빈도를 고려하여, 의료기기 분야만의 코딩룰을 제시하려 한다. 그 후에는, MISRA-C만으로는 해결할 수 없었던 오류를 위해 CWE같은 다른 Rule을 참고할 수 있을 것이다. 또한 향후에는 MISRA-C:2012를 기준으로 완성된 코딩룰을 발전시킬 수 있을 것이라 생각된다.

참고문헌

[1] 오영배, 정영은, 신석규 “의료기기 소프트웨어 기능 안전성 기술 및 표준화 동향”, TTA Journal, Vol.147, pp.87-94, 2013.

[2] Jetley, R. P., Anderson, P., Jones, P. L. “Static Analysis of Medical Device Software using CodeSonar”, ACM: In Proceedings of the 2008 workshop on Static analysis, pp.22-29, 2008.

[3] Ganesan, D., Lindvall, M., Cleaveland, R., Jetley, R., Jones, P., Zhang, Y “Architecture reconstruction and analysis of medical device software”, In Proceedings of 2011 9th Software Architecture (WICSA) Working IEEE/IFIP Conference, pp.194-203, 2011.

[4] 정다혜, 안소진, 최진영 “임베디드 SW 개발 시 프로그래밍 개선점”, 한국정보과학회, 한국컴퓨터종합학술대회논문집, Vol.39, No.1, pp.238-240, 2012.

[5] 황종규, 조현정 “열차제어시스템 소프트웨어 안전성 확인을 위한 코딩규칙 테스팅 자동화 도구의 개발”, 한국철도학회, 한국철도학회 논문집, Vol.12, No.1, pp.81-87, 2009.

[6] 노정현, 이종민, 박유현 “CodeSonar를 이용한 지역 SW 개발 업체의 결함 유형분석”, 한국정보통신학회, 한국정보통신학회논문지, Vol.19, No.3, pp.683-688, 2015.

[7] 표창우, 한경숙 “안전한 C 프로그램을 위한 코딩 표준”, 한국정보과학회, 정보과학회지, Vol.28, No.2, pp.48-54, 2010.