이벤트 기반 임베디드 소프트웨어를 위한 자동화 테스팅 기법: LG전자 오븐 제어 소프트웨어 사례 연구

박용배* 홍신* 김문주* 조준희+ 이동주+ 장훈+

*KAIST 전산학과
대전광역시 유성구 구성동 373-1

+LG 전자
서울특별시 서초구 양재대로11길 19

{yongbae.park, hongshin}@kaist.ac.kr moonzoo@cs.kaist.ac.kr {junhee.cho, dongju81.lee, bill.jang}@lge.com

요약: 이벤트 기반의 임베디드 소프트웨어는 이벤트 발생 시점이 비결정적이기 때문에 동시성 오류가 발생할 수 있다. 본 논문에서는 이벤트 발생 시점을 체계적으로 변경하여 임베디드 소프트웨어의 가능한 모든 실행을 탐색하는 이벤트 생성 프레임워크를 소개하며, 이를 이용하여 LG전자 오븐에서 사용하는 CircularQueue 모듈의 동시성 오류를 찾아내었다. 그리고 랜덤 테스팅과의 비교를 통해서 이벤트 생성 프레임워크의 효과성과 효율성을 실험으로 보였다.

핵심어: 자동화 테스팅, 동시성 오류, 임베디드 소프트웨어

1. 서 론

임베디드 소프트웨어가 다양한 분야에서 사용되면서 그 중요성이 증가하고 있으며, 임베디드 소프트웨어 시장은 매년 9.1% 이상 고속으로 성장하고 있다 [1]. 이에 따라서 임베디드 소프트웨어의 품질 또한 중요해지고 있다.

이벤트 기반의 임베디드 소프트웨어는 입력을 처리하기 위한 이벤트 핸들러와 복잡한 계산 및 출력을 처리하는 메인 루프로 구성되어 있다. 메인 루프가 무한히 반복 실행되는 도중에 외부 신호(e.g. 사용자 입력)에 의해서 이벤트가 발생하면, 메인 루프의 실행이 일시 정지되고 이벤트 핸들러가 실행된 후에 다시 메인 루프가 실행된다.

이러한 이벤트 기반 임베디드 소프트웨어에서는 다중 스레드가 없더라도 동시성 오류가 발생할 수 있다. 이벤트의 발생이 비결정적(non-deterministic)이므로 메인 루프 실행 중 어느 시점에서든지 이벤트가 발생할 수 있으며, 이벤트 핸들러는 전역 변수의 값을 바꿀 수 있기 때문이다. 즉 메인 루프가 실행되는 동안 개발자가 생각하지 못한 위치에서 이벤트가 발생하여 전역 변수가 변경되면 동시성 오류가 발생할 수 있다.

본 논문에서는 동시성 오류를 효과적으로 찾기 위해서 메인 루프 실행 중 이벤트의 발생 시점을 체계적으로 제어하여 메인 루프와 이벤트 핸들러의 가능한 모든 실행 시나리오를 실행하는 이벤트 생성 프레임워크를 소개하고, 이벤트 생성 프레임워크로 LG전자의 오븐 제어 소프트웨어의 동시성 오류를 미리 발견하여 소프트웨어의 품질을 향상시킨 사례 연구에 대해서 설명한다.

이벤트 생성 프레임워크는 메인 루프 내에서 이벤트가 발생할 수 있는 부분마다 탐침(probe)을 삽입하고, 심볼릭 변수를 이용하여 이벤트 핸들러를 실행할 탐침을 결정한다. 이벤트 생성 프레임워크는 이벤트 발생 시점을 체계적으로 변경하기 위해서 concolic 테스팅 기법 [2]을 사용한다.

사례 연구에서 LG전자 오븐의 CirqularQueue 모듈에서 2개의 동시성 오류를 찾았으며, 이벤트 생성 프레임워크와 노이즈 기반 랜덤 테스팅을 비교하는 실험을 수행한 결과, 이벤트 생성 프레임워크가 노이즈 기반 랜덤 테스팅보다 동시성 오류를 효과적으로 탐지하는 것을 보였다.

2. 관련 연구

임베디드 소프트웨어의 이벤트 핸들러로 발생할 수 있는 동시성 오류를 찾기 위해 랜덤 테스팅 기법 [3]이 제안되었으며, 이 기법은 이벤트가 발생하는 시점을 랜덤으로 결정하는 방법이다. 이 기법은 오류를 재현하는 이벤트 발생 시점을 찾을 확률이 낮은 경우에는 오류를 찾지 못하는 문제점이 있으나, 이벤트 생성 프레임워크는 이벤트 발생 시점을 체계적으로 변경하기 때문에 랜덤 테스팅보다 효과적이다.

본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업(NIPA-2013-H0301-13-5004), 한국연구재단의 중견연구자지원사업-핵심연구(NRF-2012R1A2A2A01046172), 미래창조과학부의 2013년 정보통신·방송(ICT) 연구개발사업, LG전자 SW역량강화센터의 지원으로 수행되었음.

표 1 이벤트 생성 프레임워크와 노이즈 기반 랜덤 테스팅의 버그 탐지 결과

	이벤트 생성 프레임워크	노이즈 기반 랜덤 테스팅		
		500μs	1000μs	1500μs
Overwriting 버그	1.00	1.00	1.00	1.00
Inconsistency 버그	1.00	0.20	0.23	0.00

그리고 다중 스레드 프로그램에 사용하는 모델 검증 기법을 이벤트 기반 임베디드 소프트웨어에 적용하기 위한 방법 [4]이 연구되었고, 모델 검증 기법을 적용한 사례 연구 [5]가 있다. 그러나 이 방법은 개발자가 모델을 만들어야 하기 때문에 테스팅에 많은 비용 필요하다는 단점이 있다. 반면 이벤트 생성 프레임워크에서는 탐침 삽입과 concolic 테스팅이 자동화되어 있으므로 개발자의 노력이 적게 든다.

3. 이벤트 생성 프레임워크

이벤트 생성 프레임워크는 메인 루프의 실행 도중에 이벤트가 발생할 수 있는 모든 경우의 수를 확인하기 위해서 메인 루프의 각 구문(statement) 앞에 탐침을 삽입하고, 삽입된 탐침 중 이벤트 핸들러를 호출할 탐침을 선택한다. 그림 1은 탐침이 삽입된 예제 코드이다. 탐침은 고유한 정수 값인 ID가 파라미터인 함수 호출 구문(`probe(0)`와 `probe(1)`)이며, 메인 루프 실행 중에 탐침이 실행되었을 때, ID(`locId`)와 심볼릭 변수(`evLoc`)가 동일할 경우에만 이벤트 핸들러(`handler()`)를 호출한다. 이후 탐침이 삽입된 코드를 concolic 테스팅 도구인 CREST [6]를 사용하여 실행하면 CREST는 매 실행마다 심볼릭 변수의 값을 변경하여, 각 실행마다 이벤트가 발생하는 위치를 변경하면서 테스팅을 수행한다.

4. 사례 연구

CircularQueue(CQ)는 LG전자 오븐 제어 소프트웨어에서 버튼을 통해서 들어온 사용자 입력을 저장하기 위한 모듈이다. 버튼 이벤트가 발생하면, 이벤트 핸들러는 눌린 버튼의 종류를 파악하여 `enqueue()` 함수를 통해서 CQ에 버튼 종류 정보를 저장한다. 메인 루프는 `dequeue()` 함수를 통해서 CQ에 저장된 버튼 정보를 꺼내어 사용자 입력을 처리한다. 따라서 메인 루프에서 `dequeue()`를 실행하는 도중에 이벤트 핸들러에 의해서 `enqueue()`가 호출될 수 있다.

이벤트 생성 프레임워크와 노이즈 기반 랜덤 테스팅를 CQ에 적용한 결과, CQ에서 원소의 값이 덮어씌어지는 overwrite 버그와 저장된 원소를 모두 잃어버리는 inconsistency 버그를 발견하였다. 두 버그는 `dequeue()`가 실행되는 도중에 이벤트 핸들러가 실행되어 `enqueue()`가 호출될 때 발생한다.

표 1은 이벤트 생성 프레임워크와 노이즈 기반 랜덤 테스팅을 30회 반복 수행하였을 때의 버그 탐지 비율을 나타낸다. 랜덤 테스팅은 overwriting 버그는 잘 찾지만 inconsistency 버그를 효과적으로 찾지 못한다. 이는 특정 지점에서 이벤트가 2회 연속으로 발생할 경우에만 나타나는 inconsistency 버그는 탐지할 확률이 매우 낮기 때문이다. 반면, 이벤트 생성 프레임워크는 이벤트 발생 시점을 체계적으로 선택하여 오류가 발생하는 시나리오를 생성하기 때문에 두 버그를 모두 찾아내었다.

```
1  int evLoc; // symbolic variable
2  void main() {
3    while(…) { // the main loop
4      probe(0); stmt1;
5      probe(1); stmt2;
6  } }
7  void handler() { … }
8  void probe(int locId) {
9    if(locId == evLoc) handler(); }
```

그림 1 탐침이 삽입된 코드

5. 결론 및 향후 연구

본 논문에서는 메인 루프가 실행되는 도중에 이벤트가 발생하는 시점을 체계적으로 변경하여 임베디드 소프트웨어의 동시성 오류를 찾는 이벤트 생성 프레임워크를 제안하였다. 그리고 사례 연구를 통해서 실제 소프트웨어의 버그를 조기에 찾아 고침으로써 제품의 품질을 효과적으로 향상시킬 수 있었다. 마지막으로 실험을 통해 랜덤 테스팅보다 효과적이고 효율적임을 확인하였다. 향후 연구로는 큰 프로그램에서도 이벤트 생성 프레임워크로 동시성 오류를 빠르게 찾기 위한 탐색 방법을 연구할 계획이다.

참고문헌

[1] 전자부품연구원, 임베디드 SW 경쟁력 강화방안 연구, Dec., 2013.

[2] K. Sen, D. Marinov, G. Agha, CUTE: A Concolic Unit Testing Engine for C, ESEC/FSE, 2005.

[3] J. Regehr, Random Testing of Interrupt-driven Software, EMSOFT, 2005.

[4] J. Regehr, Thread Verification vs. Interrupt Verification, ENTCS, vol. 174, no. 9, pp. 139-150, 2007.

[5] G. J. Holzmann, M. H. Smith, "A practical method for verifying event-driven software," ICSE, 1999.

[6] CREST, http://code.google.com/p/crest/.