마르코프 체인 모델을 이용한 임베디드 시스템 신뢰도 측정

곽동규°, 조용윤, 박호병, 유재우
숭실대학교 컴퓨터학과
{coolman°, yycho, r5me}@ss.ssu.ac.kr, cwyoo@comp.ssu.ac.kr

Embedded System Reliability Measurement Use Markov Chain Model

Dong-Gyu Kawk°, Yong-Yoon Cho, Ho-Byung Park, Chea-Woo Yoo
Dept of Computing, Soonsil University

요 약

임베디드 시스템은 다수의 디바이스를 컨트롤하여 시스템의 목적을 수행한다. 최근 임베디드 시스템의 요구사항이 증가함에 따라 하나의 임베디드 소프트웨어가 컨트롤하는 디바이스의 종류가 다양해지고 수도 증가하는 추세이다. 다수의 디바이스를 가지고 있는 임베디드 시스템에서 시스템의 신뢰도는 각 디바이스의 신뢰도에 많은 영향을 받는다. 본 논문은 임베디드 시스템의 신뢰도를 측정하기 위해서 통계적 신뢰도 측정 방법 중 한 가지인 마르코프 체인을 이용한 방법을 제안한다. 마르코프 체인은 여러 분야에서 복잡한 시스템을 단순화하여 모델링하고 과거의 변화를 토대로 미래를 예측할 수 있는 방법을 제공한다. 또한, 전체 시스템의 확률을 행렬로 계측할 수 있는 방법을 가지고 있어 특정 부분의 확률이 전체 시스템의 확률에 미치는 영향을 산술적으로 계산할 수 있는 장점을 가지고 있다.

본 논문에서 제안하는 임베디드 소프트웨어 마르코프 체인은 테스트 대상 소스를 분석하여 디바이스를 컨트롤하는 루틴과 에러를 핸들링하는 루틴, 일반적인 루틴으로 나누어 각각을 상태로 정의한다. 정의한 각 상태간의 전이는 통계적으로 측정한 디바이스 신뢰도를 확률로 표현한다. 마르코프 체인을 이용하여 임베디드 시스템의 신뢰도를 측정하기 위한 시스템은 소스 분석기와 신뢰도 측정기로 나누어 설계한다. 소스 분석기는 테스트 대상이 되는 소스와 디바이스 드라이버 라이브러리 테이블을 입력으로 하고 소프트웨어의 마르코프 체인을 출력으로 한다. 마르코프 체인은 행렬로 표현하고 연산하여 시스템의 신뢰도를 측정한다. 제안하는 시스템의 신뢰도 측정 방법은 부분이 가지고 있는 신뢰도가 전체 신뢰도에 미치는 영향을 산술적으로 측정할 수 있어 시스템이 요구하는 신뢰도에 접근할 수 있는 방법과 근거를 제공하는 장점이 있다.

1. 서 론

임베디드 시스템이란 특정 기능을 수행하기 위해 컴퓨터가 어떤 시스템에 내장되어 사용되는 방식으로 컴퓨터가 제품이나 시스템의 일부분에 통합된 것이다. 임베디드 시스템은 일반적으로 시스템이 많은 디바이스를 컨트롤하며 동작한다. 즉, 일반적인 소프트웨어에 비해 디바이스와 디바이스 드라이버의 신뢰도가 전체 소프트웨어의 신뢰도에 많은 영향을 미친다. 본 논문은 신뢰도 측정 모델 중 하나인 마르코프 체인 모델[1]을 이용하여 임베디드 소프트웨어와 디바이스 신뢰도를 측정하는 방법을 제안한다.

마르코프 체인은 많은 분야에서 시스템을 모형화하고 시스템의 변화를 예측하는데 사용되고 있다. 이 방법은 부분의 확률로 전체 확률을 측정할 수 있는 장점을 가지고 있어 실험이 어려운 무한 횟수에 대한 상태를 예상할 수 있는 장점을 가지고 있다. 소프트웨어 테스팅 분야에서는 블랙박스 테스팅의 한 방법으로 각 소프트웨어 상태에서 전의되는 확률을 통계적으로 측정하고 상태의 추이과정을 마르코프 체인으로 도식화하여 신뢰도를 측정하는 방법에 대한 연구가 있었다[2][3]. 하지만 이 방법은 소프트웨어의 신뢰도 향상을 위한 정보를 제공할 수 없어 높은 신뢰도의 소프트웨어를 생산하기 위해서는 다른 정보를 이용하여야 한다.

본 논문에서 제안하는 신뢰도 측정 방법은 디바이스를 컨트롤하는 임베디드 소프트웨어를 대상으로 문장의 역할에 따라 몇 개의 문장을 하나의 상태로 정의하고 각 상태에서 다른 상태로 전이 하는 과정을 확률로 측정한다. 일반적으로 디바이스를 컨트롤하는 소프트웨어의 코드는 디바이스 드라이버를 획득하기 위한 변수의 초기화 루틴과 디바이스 드라이버 획득, 드라이버 컨트롤, 드라이버 해제로 이루어진다. 이 때 디바이스 드라이버를 획득과 컨트롤하는 루틴은 디바이스가 정상적으로 동작할 경우와 동작하지 않을 경우로 나누어 프로그램을 작성한다. 제안하는 시스템은 임베디드 소프트웨어의 신뢰도를 측정하기 위해서 소스 중 디바이스를 컨트롤하는 문장과 일반적인 프로그램을 실행 하는 문장, 오동작을 처리하는 문장으로 나누어 분석하고 각각을 상태로 정의한다. 또한, 각 상태에서 다음 상태로 전이하는 확률을 통계적으로 측정하여 마르코프 체인을 작성하고, 전체 시스템의 신뢰도를 측정한다. 본 시스템은 각각 디바이스의 신뢰도가 전체 시스템의 신뢰도에 미치는 영향을 산술적으로 계산할 수 있는 장점을 가지고 있다.

본 논문은 2장에서 관련 연구를 보이고 3장에서 본 시스템에서 사용한 마르코프 체인에 대해 설명한 다음 4장에서 제안하는 시스템의 구조를 보인 후 5장에서 결론과 향후 연구 과제를 보인다.

2. 관련 연구

2.1 마르코프 체인(Markov Chain)[1]

마르코프 체인이란 한 상태에서 다른 상태를 전이할 때의 확률이 이전에 상태에만 영향을 받을 경우 이 전체 상태를 말한다. 마르코프 체인의 수학적인 정의는 다음과 같다.

○ 마르코프 체인

$X(t)$가 확률과정일 때 $X(t)$가 취하는 값을 상태라 한다.

$X(n)=a_i$인 것을 n단계(n회)시행에서 상태 a_i에 있다고 한다. 따라서 확률변수 $X(t)$가 $X_1, X_2, \ldots, X_n$일 때 대응하는 상태공간은 유한, 가산집합 $a_1, a_2, \ldots, a_n$에 속한다. 상태의 순서쌍(a_i, a_j)의 확률은 $P_{ij}^{(n)}$으로 나타내고 확률변수 $X(n)$, $X(n+1)$의 상태가 a_i, a_j일 때 상태 a_i의 바로 다음 단계의 시행결과의 확률이 $P_{ij}^{(n)}$이라는 뜻이다. 이러한 추이과정을 마르코프 체인이라 한다.

마르코프 체인은 어떤 시스템을 모형화하는 수학적 기법으로 과거에 있었던 변화를 토대로 시스템의 여러 변수들이 갖고 있는 동적 성격을 파악하여 미래에 있을 변화를 연속적으로 예측하는데 사용되고 있다.

2.2 A Markov Chain Model for Statistical Software Testing[2]

James는 블랙박스 소프트웨어의 테스팅을 통계적으로 예측하기 위해 마르코프 체인을 이용하였다. 이 방법에서는 소프트웨어의 상태를 사용자의 입력으로, 상태 전이 확률을 사용자의 입력 확률로 정의하였다. 또한, 정의한 상태와 확률을 이용하여 마르코프 체인을 작성하고 작성된 마르코프 체인을 이용하여 신뢰도를 측정하는 방법을 제시하였다. 이 방법은 테스팅하기 어려운 전체 시스템을 테스트하지 않고 부분의 통계적인 확률 테스팅만으로 전체 시스템의 신뢰도를 측정할 수 있는 장점을 가지고 있다. 하지만, James는 GUI에 대한 실험만을 실시하였고 소스 코드의 분석이 없는 블랙박스 테스트만을 시행하여 신뢰도를 증가시키기 위한 방법을 제시하지 못하였다.

3. 임베디드 소프트웨어 마르코프 체인 모델

본 논문은 임베디드 시스템에서 디바이스의 신뢰도가 전체 소프트웨어에 미치는 영향을 측정하기 위해서 마르코프 체인을 임베디드 소프트웨어에 특정화 시켜 임베디드 소프트웨어 마르코프 체인을 제안한다.

● 임베디드 소프트웨어 마르코프 체인 모델

상태 집합 $S=N \cup D \cup E$

(단, N은 일반 루틴, D는 디바이스 컨트롤 루틴, E 에러 핸들링 루틴, $N \cap D=\phi, D \cap E=\phi, E \cap N=\phi$)

$S(n)=S_i$ 일 때, 순서쌍 (S_i, S_j)는 S_i상태에서 S_j상태로 정이하는 확률이고 $P_{ij}^{(n)}$으로 표현한다.

표 1은 LED를 컨트롤하는 간단한 예제로써 n회 동안 LED를 컨트롤하는 소프트웨어이다.

표 1 디바이스 컨트롤 예제

```
int LED_Control(){
  int i, LED_fd; // N1 start
  unsigned char *mmap_addr;
  unsigned char *LEDReg; // N1 end
  if ((LED_fd = open("/dev/mem", O_RDWR|O_SYNC)) < 0){ // D1 start
end
    perror(" Fail: File Descriptor isn't Opend \n"); // E1 start
    exit(2); // E1 end
  }
  mmap_addr = mmap(NULL,1024,(PROT_READ|PROT_WRITE),
MAP_SHARED,
  LED_fd, 0x08000000); // D2 start end
  if (mmap_addr < 0) {
    mmap_addr = NULL; // E2 start
    printf(" Fail: Memory mapping error\n");
    return -1; // E2 end
  }
  LEDReg = (unsigned char *)(mmap_addr + 8); // N2 start
  i = 0; // N2 end
  while(i<8){
    i++;
    *LEDReg = 0x01 << i ; // N3 start
    sleep(1); // N3 end
  }
  munmap(mmap_addr, 1024); // N4 start
  close(LED_fd);
  return 0; // N4 end
}

int main(){
  int i;
  for(i = 0; i < n; i++){
    LED_Control();
  }
  return 0;
}
```

표 1의 예제는 네 개의 일반 루틴과 두 개의 디바이스 컨트롤 루틴, 두 개의 에러 핸들링 루틴으로 이루어져 있다. 각 루틴을 상태로 정의하고 상태의 전이 확률을 표현하여 전이 확률이 1인 상태를 통합하여 표현하면 그림 1과 같다.

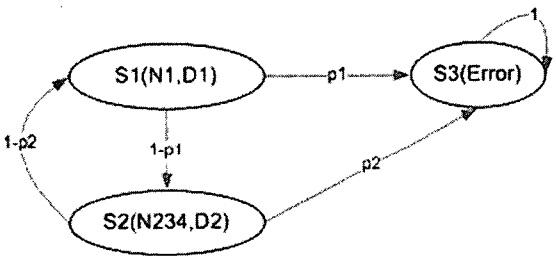

그림 1 간단한 마크코프 체인 모델 예제

그림 1에서 P_1과 P_2는 각각 상태D_1과 D_2가 상태 E_1과 E_2로 전이할 확률로써 각 디바이스를 통계적인 에러 확률이다. 아래 행렬은 그림 1의 마르코프 체인 모델에 따라 행렬로 작성한 것이다.

전이행렬 $T=\begin{pmatrix} 0 & 1-p_1 & p_1 \\ 1-p_2 & 0 & p_2 \\ 0 & 0 & 1 \end{pmatrix}$이다. T^n을 이용하여 n회 시행 시, D_1에서 E로 전이할 확률과 D_2에서 E로 전이

할 확률을 각각 $P_1^{(n)}$과 $P_2^{(n)}$이라 하면, $P_1^{(n)}$과 $P_2^{(n)}$는 아래와 같다.

$$P_1^{(n)} = \begin{cases} p_1 & n=1 \\ \sum_{k=1}^{\frac{n-1}{2}} ((1-p_1)^k(1-p_2)^{k-1}(p_1(1-p_2)+p_2)) + p_1 & n \bmod 2 = 1, n>1 \\ \sum_{k=1}^{\frac{n}{2}} ((1-p_1)(1-p_2))^{k-1}((1-p_1)p_2+p_1) & n \bmod 2 = 0, n>1 \end{cases}$$

$$P_2^{(n)} = \begin{cases} 0 & n=1 \\ \sum_{k=1}^{\frac{n-1}{2}} ((1-p_1)^{k-1}(1-p_2)^k(p_2(1-p_1)+p_1)) + p_2 & n \bmod 2 = 1, n>1 \\ \sum_{k=1}^{\frac{n}{2}} ((1-p_1)(1-p_2))^{k-1}((1-p_2)p_1+p_2) & n \bmod 2 = 0, n>1 \end{cases}$$

위 확률을 $p_1 \approx p_2 \approx p$로 단순화 하고 상한 값을 계산하면 $O(p^n)$이다. 그러므로 p의 값이 m이 감소하면 $P^{(n)}$은 $O(m^n)$감소하는 효과를 가져 올수 있다. 또한, 여분의 디바이스가 동작하거나 디바이스 에러를 복원이 q의 확률로 재작동하는 경우 신뢰도를 향상할 수 있다. 그림 2는 q의 확률로 디바이스가 복원되는 경우의 마르코프 체인 모델이다.

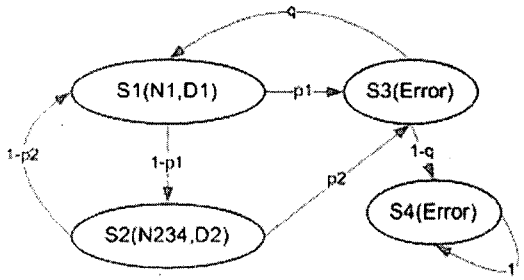

그림 2 복원 확률을 적용한 마크코프 체인

아래 행렬은 그림 2에 대한 마르코프 체인의 전이행렬이다.

전이행렬 $T = \begin{pmatrix} 0 & 1-p_1 & p_1 & 0 \\ 1-p_2 & 0 & p_2 & 0 \\ q & 0 & 0 & 1-q \\ 0 & 0 & 0 & 1 \end{pmatrix}$이고, 따라서, n회 시행하는 경우, 상태 S_4 즉, 에러로 갈 확률이 계수$(1-q)$를 갖고, $O(n)$으로 감소한다.

4. 시스템 구조

제안하는 신뢰도 측정 시스템은 테스트 대상 소스를 분석하는 소스 분석기와 분석된 마르코프 체인 모델을 행렬로 연산하는 신뢰도 측정기로 나누어 설계한다. 그림 3은 본 시스템의 구조도이다.

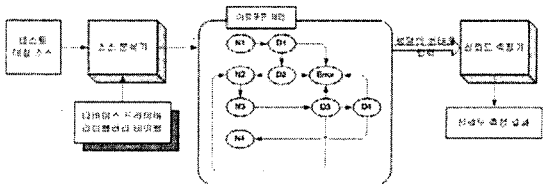

그림 3 신뢰도 측정 시스템 구조도

소스 분석기는 테스트 대상 소스를 분석하여 소스의 각 문장을 세 가지 형태로 분류한다. 문장을 각 형태로 분류하는 정보는 디바이스 드라이버 라이블러리 테이블에 존재하고 이 테이블은 각 디바이스의 통계적인 신뢰도를 포함되어 있다. 마르코프 체인은 전이행렬로 표현하고 신뢰도 측정기를 통해 전이행렬의 n승으로 n회 실행에 대한 신뢰도를 측정한다.

5. 결론 및 향후 연구 과제

임베디드 시스템이 복잡해짐에 따라 시스템의 신뢰도 측정에 대한 요구가 증가하고 있다. 임베디드 시스템는 특성상 디바이스의 컨트롤로 동작하고 각 디바이스의 신뢰도가 전체 임베디드 시스템의 신뢰도에 많은 영향을 미친다. 본 논문은 통계적으로 시스템의 상태를 측정할 수 있는 방법 중 하나인 마르코프 체인을 이용한 임베디드 시스템 신뢰도 측정 방법을 제안한다. 기존의 소스를 분석하지 않고 테스팅을 실시하는 블랙박스 테스팅은 신뢰도를 증가시키기 위한 정보를 제공하고 있지 않아 신뢰도 증가에 어려움이 있었다. 본 시스템은 소스를 분석하여 각 디바이스 모듈의 신뢰도가 전체 시스템 신뢰도에 미치는 영향을 분석할 수 있어 요구하는 신뢰도에 접근할 수 있는 방법을 찾을 수 있다. 하지만, 임베디드 소프트웨어의 분기에 대한 부분을 고려하고 있지 않고 있다. 소프트웨어의 분기는 입력 값에 따라 이루어짐으로 테스트 데이터[4]를 생성하여 통계적으로 측정하여야 하고 생성하는 마르코프 체인의 복잡도도 증가한다. 본 시스템은 다수의 디바이스를 컨트롤하는 임베디드 시스템의 신뢰도 향상에 기여할 것으로 기대된다.

참고 문헌

[1] 이은구, "Markov Chain에 관한 연구", *수학교육학회지*, 20권 3호, pp.19-22, 1982.6.

[2] J. Whittaker and M. Thomason, "A Markov Chain Model for Statistical Software Testing", *IEEE Trans. Software Engineering*, vol 20. no 10, pp 812-824, 1995.

[3] Rajgopal, J. Mazumdar, M. , "Modular operational test plans for inferences on software reliability based on a Markov model", *Software Engineering, IEEE Transactions*, vol 28, pp.358-363, 2002. 4.

[4] 정인상, "소프트웨어 테스팅을 위한 테스트 데이터의 자동 생성" 정보과학회지 제19권 제11호, pp.10-18, 2001. 11.