



(19) 대한민국특허청(KR)  
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0127334  
(43) 공개일자 2015년11월17일

(51) 국제특허분류(Int. Cl.)

G06F 11/30 (2006.01) G06F 11/07 (2006.01)

(21) 출원번호 10-2014-0053937

(22) 출원일자 2014년05월07일

심사청구일자 2014년05월07일

(71) 출원인

현대자동차주식회사

서울특별시 서초구 현릉로 12 (양재동)

이화여자대학교 산학협력단

서울특별시 서대문구 이화여대길 52 (대현동, 이화여자대학교)

기아자동차주식회사

서울특별시 서초구 현릉로 12 (양재동)

(72) 발명자

최병주

서울특별시 강남구 압구정로 151 신현대아파트 7  
4동 1305호

오정훈

경기도 용인시 수지구 성북2로76번길 31 대우푸르  
지오 105동 801호

(뒷면에 계속)

(74) 대리인

한라특허법인

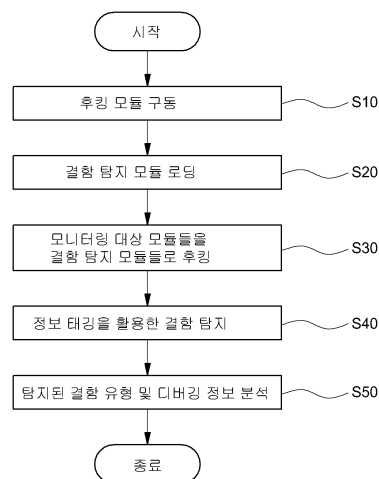
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 실행 후킹과 정보 태깅을 이용하는 런-타임 결함 탐지 방법

### (57) 요약

런-타임 결함 탐지 방법 및 런-타임 결함 탐지 프로그램이 기록된 기록매체가 개시된다. 본 발명의 일 실시예에 따른 런-타임 결함 탐지 방법은, (a) 모니터링 대상 모듈들에 대응하며 정보 태깅 기능 및 결함 탐지 기능이 추가된 결함 탐지 모듈들을 로딩시키는 단계; (b) 상기 모니터링 대상 모듈들의 실행 경로를 상기 결함 탐지 모듈들로 후킹하는 단계; 및 (c) 상기 결함 탐지 모듈들의 실행에 따라 생성되는 데이터에 정보 태그를 태깅하고 상기 데이터의 사용 시점에서 상기 정보 태그를 검사하여 런-타임 결함을 탐지하는 단계;를 포함한다.

### 대표도 - 도3



(72) 발명자

**장승연**

경기도 화성시 남양성지로 203-9 선주하이츠빌 C동  
204호

**양승환**

경기도 군포시 금산로 91 래미안하이어스아파트  
117동 1202호

**서주영**

서울특별시 송파구 중대로 24 올림픽헤밀리타운  
105동 304호

## 명세서

### 청구범위

#### 청구항 1

- (a) 모니터링 대상 모듈들에 대응하며 정보 태깅 기능 및 결함 탐지 기능이 추가된 결함 탐지 모듈들을 로딩시키는 단계;
- (b) 상기 모니터링 대상 모듈들의 실행 경로를 상기 결함 탐지 모듈들로 후킹하는 단계; 및
- (c) 상기 결함 탐지 모듈들의 실행에 따라 생성되는 데이터에 정보 태그를 태깅하고 상기 데이터의 사용 시점에서 상기 정보 태그를 검사하여 런-타임 결함을 탐지하는 단계;를 포함하는 런-타임 결함 탐지 방법.

#### 청구항 2

제1항에 있어서,

상기 (c) 단계는,

- (c1) 상기 결함 탐지 모듈들의 실행에 따라 생성되는 데이터에 정보 태그를 태깅하는 단계;
- (c2) 상기 데이터의 사용 시점에서 상기 태그 정보를 검사하여 런-타임 결함을 탐지하는 단계;
- (c3) 탐지된 런-타임 결함의 유형 및 디버깅 정보를 기록하는 단계; 및
- (c4) 모니터링 대상 모듈의 본래 기능을 수행하는 단계;를 포함하는 런-타임 결함 탐지 방법.

#### 청구항 3

제1항에 있어서,

- (d) 탐지된 런-타임 결함의 유형과 디버깅 정보를 분석하고, 분석 결과를 사용자에게 제공하는 단계;를 더 포함하는 런-타임 결함 탐지 방법.

#### 청구항 4

제3항에 있어서,

상기 (a), (b), 및 (c) 단계는 모니터링 대상 시스템에서 수행되고, 상기 (d) 단계는 상기 모니터링 대상 시스템과 유선 또는 무선으로 통신하는 호스트 기기에서 수행되는 런-타임 결함 탐지 방법.

#### 청구항 5

제1항에 있어서,

상기 정보 태그에는 데이터 주소 공간, 데이터 초기화 여부, 및 주소 공간의 유효 사이즈가 기록되는 런-타임 결함 탐지 방법.

#### 청구항 6

런-타임 결함 탐지 프로그램이 기록된 기록 매체에 있어서,

상기 런-타임 결함 탐지 프로그램은,

모니터링 대상 모듈들에 대응하며, 정보 태깅 기능 및 결함 탐지 기능이 추가된 결함 탐지 모듈들;

상기 결함 탐지 모듈들을 로딩시키는 컴포넌트 관리 모듈; 및

상기 모니터링 대상 모듈들의 실행 경로를 상기 결함 탐지 모듈들로 후킹하는 후킹 모듈;을 포함하며,

상기 결함 탐지 모듈들은, 상기 결함 탐지 모듈들의 실행에 따라 생성되는 데이터에 정보 태그를 태깅하고 상기 데이터의 사용 시점에서 상기 정보 태그를 검사하여 런-타임 결함을 탐지하는 수행하는, 런-타임 결함 탐지 프로그램이 기록된 기록 매체.

## 청구항 7

제6항에 있어서,

상기 런-타임 결함 탐지 프로그램은, 탐지된 런-타임 결함의 유형과 디버깅 정보를 분석하고 분석 결과를 사용자에게 제공하는 분석 모듈을 더 포함하는, 런-타임 결함 탐지 프로그램이 기록된 기록 매체.

## 발명의 설명

### 기술 분야

[0001] 본 발명은 런-타임 결함 탐지 방법에 관한 것으로서, 보다 구체적으로는, 실행 후킹(execution hooking)과 정보 태깅(information tagging)을 이용하는 런-타임 결함 탐지 방법에 관한 것이다.

### 배경 기술

[0002] 런-타임 결함이란 프로세스가 실행되는 동안 발생하는 결함을 의미하며, 시스템을 다운시키거나 프로그램 실행을 중단시키기도 한다. 이러한 런-타임 결함을 분석하기 위하여 일반적으로 결함이 발생한 순간의 데이터를 수집하여 결함을 재현하려 시도한다.

[0003] 실행 순간을 결정짓는 데이터에 있어서, 일반적인 시스템의 경우에는 사용자 입력의 비중이 큰 반면, 임베디드 시스템(embedded system)의 경우에는 사용자 입력뿐만 아니라 외부 환경으로부터 입력되는 데이터도 다양하다. 예를 들어 차량에 탑재되는 AVN(Audio Video Navigation) 시스템의 경우 위성센터로부터 수신되는 GPS 좌표 값, 방송국으로부터 수신되는 라디오 주파수, 교통 상황을 알려주는 TPEG 신호와 같이 외부 환경으로부터 다양한 입력이 있게 되며, 그러한 외부 입력들은 시스템의 실행에 중요한 영향을 미친다. 이처럼 임베디드 시스템에서는 실행을 결정짓는 데이터가 다양하기 때문에 발생 결함과 관련된 데이터를 수집하는 것도 너무 광범위한 일이며 이를 토대로 결함을 재현하는 것도 쉽지 않다.

[0004] 한편, 런-타임 결함을 재현하기 위해 결함이 발생한 순간의 시스템 상태를 덤핑(dumping)하는 기법과 실행 히스토리를 저장했다가 롤백(rollback)을 통해 역추적하는 기법들이 많이 이용된다.

[0005] 덤핑 정보는 시스템이 다운될 때 그 순간의 시스템의 런-타임 상태를 분석하는데 도움이 된다. 하지만 일반적으로 시스템이 다운되는 시점이 실제 결함을 야기한 시점과 일치하지 않는 경우가 더 많으므로, 덤핑 정보만으로는 디버깅(debugging)을 위한 소스 코드의 위치를 파악하기가 쉽지 않다.

[0006] 그리고, 결함 위치를 추적하려면 실행 히스토리를 저장해야 하는데, 특히 임베디드 시스템의 경우에는 외부 입력에 의해 실행이 결정되는 경우가 많기 때문에 저장해야 할 데이터의 규모도 방대할 뿐 아니라, 결함을 야기한 시점을 알 수 없기 때문에 실행 히스토리를 어느 정도의 깊이로 저장해야 할지를 결정하기 어렵다.

[0007] 또한, 시스템이 다운되지 않고 데드락(deadlock) 상태이거나 메모리 누수(memory leak)와 같이 결함 증상이 겉으로 드러나지 않는 잠재 결함의 경우에는 아예 덤핑 정보가 생성되지 않게 되는 한계도 있다. 따라서 시스템 다운이 발생하지 않는 경우 런-타임 결함에 대한 디버깅은 더욱 어려워진다.

## 발명의 내용

### 해결하려는 과제

[0008] 본 발명의 목적은, 결함 발생 이후에 결함을 재현하는 방식을 벗어나 시스템이 실행되는 동안 결함을 탐지하여 결함 유형과 디버깅 정보를 제공함으로써 런-타임 결함 탐지의 정확성을 높이면서도 그에 소요되는 노력을 줄일 수 있는 방안을 제공하는 것이다.

[0009]

### 과제의 해결 수단

[0010] 상기 목적을 달성하기 위해, 본 발명은, (a) 모니터링 대상 모듈들에 대응하며 정보 태깅 기능 및 결함 탐지 기능이 추가된 결함 탐지 모듈들을 로딩시키는 단계; (b) 상기 모니터링 대상 모듈들의 실행 경로를 상기 결함 탐지 모듈들로 후킹하는 단계; 및 (c) 상기 결함 탐지 모듈들의 실행에 따라 생성되는 데이터에 정보 태그를 태깅하고 상기 데이터의 사용 시점에서 상기 정보 태그를 검사하여 런-타임 결함을 탐지하는 단계;를 포함하는 런-타임 결함 탐지 방법을 제공한다.

[0011] 상기 (c) 단계는, (c1) 상기 결함 탐지 모듈들의 실행에 따라 생성되는 데이터에 정보 태그를 태깅하는 단계; (c2) 상기 데이터의 사용 시점에서 상기 태그 정보를 검사하여 런-타임 결함을 탐지하는 단계; (c3) 탐지된 런-타임 결함의 유형 및 디버깅 정보를 기록하는 단계; 및 (c4) 모니터링 대상 모듈의 본래 기능을 수행하는 단계;를 포함할 수 있다.

[0012] 상기 런-타임 결함 탐지 방법은, (d) 탐지된 런-타임 결함의 유형과 디버깅 정보를 분석하고, 분석 결과를 사용자에게 제공하는 단계;를 더 포함할 수 있다.

[0013] 상기 (a), (b), 및 (c) 단계는 모니터링 대상 시스템에서 수행되고, 상기 (d) 단계는 상기 모니터링 대상 시스템과 유선 또는 무선으로 통신하는 호스트 기기에서 수행될 수 있다.

[0014] 상기 정보 태그에는 데이터 주소 공간, 데이터 초기화 여부, 및 주소 공간의 유효 사이즈가 기록될 수 있다.

[0015] 상기 목적을 달성하기 위해, 본 발명은, 런-타임 결함 탐지 프로그램이 기록된 기록 매체로서, 상기 런-타임 결함 탐지 프로그램은, 모니터링 대상 모듈들에 대응하며, 정보 태깅 기능 및 결함 탐지 기능이 추가된 결함 탐지 모듈들; 상기 결함 탐지 모듈들을 로딩시키는 컴포넌트 관리 모듈; 및 상기 모니터링 대상 모듈들의 실행 경로를 상기 결함 탐지 모듈들로 후킹하는 후킹 모듈;을 포함하며, 상기 결함 탐지 모듈들은, 상기 결함 탐지 모듈들의 실행에 따라 생성되는 데이터에 정보 태그를 태깅하고 상기 데이터의 사용 시점에서 상기 정보 태그를 검사하여 런-타임 결함을 탐지하는 수행하는, 런-타임 결함 탐지 프로그램이 기록된 기록 매체를 제공한다.

[0016] 상기 런-타임 결함 탐지 프로그램은, 탐지된 런-타임 결함의 유형과 디버깅 정보를 분석하고 분석 결과를 사용자에게 제공하는 분석 모듈을 더 포함할 수 있다.

### 발명의 효과

[0017] 본 발명에 의하면, 결함 발생 이후에 결함 재현을 통해 결함을 분석하지 않고, 정보 태깅 기법을 사용하여 프로세스가 실행되는 런-타임에 결함 발생을 탐지하므로, 디버깅이 필요한 코드 위치 정보 탐색의 정확성을 크게 개선할 수 있을 뿐만 아니라 그 위치 정보 탐색에 드는 노력을 최소화할 수 있다.

[0018] 그리고, 본 발명에 의하면, 정보 태깅을 통해 런-타임에 결함 발생 가능성을 판단하고 그 판단의 결과로 탐지되는 결함들만을 선별적으로 디버깅 정보로 제공하는 방식을 취하므로, 웨도우 메모리 기법 등의 종래 기술에서 발생하는 메모리 공간 오버헤드(memory space overhead) 및 성능 오버헤드(performance overhead) 문제가 해소될 수 있다.

[0019] 그리고, 본 발명에 의하면, 프로세스가 실행되는 런-타임에 결함을 탐지하므로, 시스템이 다운되는 경우에 결함 탐지가 가능함은 물론, 데드락 상태 또는 메모리 누수와 같이 결함 증상이 겉으로 드러나지 않는 잠재 결함들에 대해서도 결함 탐지가 가능한 이점이 있다.

[0020] 또한, 본 발명에 의하면, 결함을 재현하는 방식을 취하지 않으므로, 사용자 입력 외에 다양한 외부 입력이 수신되는 차량용 네비게이션 시스템과 같은 임베디드 시스템에 특히 유용하게 적용될 수 있다.

### 도면의 간단한 설명

[0021] 도 1은 본 발명에 따른 정보 태그의 일 예를 설명하기 위한 도면이다.  
 도 2는 본 발명의 실시예에 따른 런-타임 결함 탐지 프로그램을 나타낸 블록도이다.  
 도 3은 본 발명의 실시예에 따른 런-타임 결함 탐지 방법을 나타낸 흐름도이다.  
 도 4는 런-타임 결함 탐지 방법의 제4 단계(결함 탐지 단계)를 구체적으로 보이는 흐름도이다.

### 발명을 실시하기 위한 구체적인 내용

[0022] 이하에서는 첨부된 도면들을 참조하여 본 발명에 대해 보다 상세히 설명하기로 한다.  
 [0023] 본 발명은 런-타임 결함의 디버깅을 위해 결함 발생 이후 결함을 재현하는 방식을 벗어나, 프로세스가 실행되는 동안에 즉 런-타임에 결함 발생 가능성을 탐지하여 결함 유형과 디버깅 포인트를 제공하는 방안을 제시한다.  
 [0024] 본 발명이 제안하는 방안은, 런-타임 결함과 관련된 프로세스 모듈들의 실행 순간을 후킹(hooking)하고, 프로세스가 실행되는 동안 런-타임으로 결함 탐지에 필수적인 정보를 데이터에 태깅(tagging)하여 그 태그 정보를 토대로 런-타임 결함을 탐지하는 방식이다.  
 [0025] 이와 같이 본 발명이 제안하는 방안은 런-타임 결함 분석을 위해 '실행 후킹'과 '정보 태깅' 기법을 사용한다는 점을 주목할 필요가 있다. 이에 '후킹'과 '정보 태깅'에 대해 소개한 후 본 발명의 실시예에 대해 구체적으로 살펴보기로 한다.

### 모니터링 시점 후킹:

[0026] '후킹'은 특정 목적을 위해 실행 타임에 시스템의 운영과 제어 방식을 가로채어 원하는 방향으로 변경하는 프로그래밍 기법을 의미하며, 프로세스의 실행 상황을 런-타임에 모니터링하기에 유용한 방식이다.  
 [0028] 일반적으로 프로세스가 실행될 때 사용되는 프로그램 모듈들의 호출은 시스템의 커널 공간(kernel space)에 속한 프로세스 제어 블록(process control block: PCB)을 통해 관리된다. 프로세스가 어떤 모듈을 실행하고자 그 모듈을 호출하면 프로세스 제어 블록(PCB)은 호출된 모듈이 저장된 주소를 제공하여 호출된 모듈이 실행될 수 있도록 한다.  
 [0029] 본 발명의 경우, 어떤 모니터링 대상 모듈의 호출이 있을 때 그 모듈의 본래 기능(original function)에 더하여 런-타임 결함 탐지를 위한 추가 기능을 가진 대체 모듈 이른바 '결함 탐지 모듈'이 대신 호출되도록, PCB 상의 모니터링 대상 모듈의 주소를 결함 탐지 모듈의 주소로 후킹하는 방식이 사용된다. 후킹에 필요한 프로세스 제어 블록 정보들은 커널의 내부 자료구조이므로, 이러한 후킹 방식이 적용되기 위해서는 커널 공간에 접근하여 호출된 모듈 주소를 후킹하는 커널 드라이버로서 '후킹 모듈(hooking module)'이 필요하다.  
 [0030] 메모리 결함 탐지를 예로 들면, 하기 [표 1]에 기재된 프로그램 모듈들이 모니터링 대상 모듈로 선정될 수 있다.

표 1

|        |              |                                    |                                                                                            |
|--------|--------------|------------------------------------|--------------------------------------------------------------------------------------------|
| [0031] | 모니터링 목적      | 모니터링 대상 모듈                         |                                                                                            |
|        | 크래쉬 포인트 모니터링 | SIGSEGV, SIGABRT, SIGFPE의 시그널 핸들러들 |                                                                                            |
|        | 디버깅 포인트 모니터링 | MEMORY                             | malloc(), calloc(), realloc(), free(), memcpy(), memmove(), memset(), memchr(), memalign() |
|        |              |                                    |                                                                                            |

|        |                                                  |
|--------|--------------------------------------------------|
| STRING | strlen(), strcpy(), strcat(), strcmp(), strchr() |
| I/O    | open(), close(), read(), write()                 |

[0032] 대부분의 운영체제들은 시스템이 크래쉬(crash)되는 런-타임 결함이 발생하는 경우 SIGSEGV, SIGABRT, SIGFPE와 같이 시스템 충돌의 원인과 연관된 시그널을 발생시키며, 담당 시그널 핸들러(signal handler)를 통해 시스템 충돌 순간의 시스템 상태 정보를 덤프하도록 설계되어 있다. 따라서, 크래쉬 포인트(crash point)를 모니터링하고자 하는 경우, [표 1]에 기재된 바와 같은 SIGSEGV, SIGABRT, SIGFPE의 시그널 핸들러들을 모니터링 대상 모듈로 선정하여 그 대상 모듈들이 호출되는 시점을 모니터링할 필요가 있다.

[0033] 결함이 실제 시작되는 디버깅 포인트(debugging point)에 대한 정보를 얻기 위해서는 시그널 핸들러가 호출될 때의 상태 정보 만으로는 부족하며 그 이전에 실행되었던 히스토리 정보가 필요하다. 따라서, 디버깅 포인트 모니터링을 위해서는 메모리 결함 안전에 취약한 것으로 알려진 [표 1]에 기재된 malloc(), calloc(), realloc(), strlen(), strcpy(), open(), close() 등의 함수들(모듈들)의 실행 시점을 모니터링할 필요가 있다.

[0034] 이와 같이 크래쉬 포인트 모니터링 목적의 대상 모듈들로는 SIGSEGV, SIGABRT, SIGFPE의 시그널 핸들러들이 선정될 수 있고 디버깅 포인트 모니터링 목적의 대상 모듈로는 malloc(), calloc(), realloc(), strlen(), strcpy(), open(), close() 등의 함수들이 선정될 수 있는데, 이러한 대상 모듈들의 실행 시점을 모니터링하기 위해 본 발명에서는 후킹 기법을 적용하여 각 대상 모듈들의 실행 시점에서 결함 탐지 기능을 추가로 가진 전술한 결함 탐지 모듈로 후킹한다. 예로써, 모니터링 대상 모듈인 'malloc()'가 호출되면 그 모듈이 실행되는 것이 아니라 malloc() 모듈에 대응하는 결함 탐지 모듈이 실행됨으로써, malloc() 모듈의 실행 시점이 모니터링될 수 있다.

#### [0035] 런-타임 정보 태깅:

[0036] 시스템 크래쉬로 덤프된 데이터를 통해서 시스템 충돌 위치, 즉 크래쉬 포인트를 알아내는 것은 가능하지만, 디버깅 포인트를 파악하기에는 충분하지 않다. 또한 시스템이 다운된 것처럼 보이지만 실제 데드락이 걸린 hang-up 상태라든지, 메모리 누수와 같이 결함의 증상이 나타나지 않고 프로세스가 정상 동작하는 것처럼 보이는 잠재 결함의 경우엔 크래쉬 덤프 파일이 생성조차 되지 않는다.

[0037] 이에 본 발명은, 디버깅 포인트를 보다 쉽게 파악할 수 있도록 하기 위해 그리고 시스템 크래쉬가 나지 않은 경우에도 디버깅 포인트를 파악할 수 있도록 하기 위해, '런-타임 정보 태깅' 방안을 제안한다. 이러한 런-타임 정보 태깅은, 데이터가 유효성을 런-타임에 판단하여 결함 발생 여부를 가려내기 위해서 원본 데이터에 '정보 태그(information tag)'라고 불리는 결함 판정에 필요한 정보가 기록되는 추가 필드를 함께 저장하는 것이다.

[0038] 정보 태그의 일 예를 도시한 도 1을 참조하면, 정보 태그(2)는 런-타임 결함 탐지에 필요한 데이터 속성이 저장되는 필드로서 원본 데이터(1)에 태깅된다. 예로써, 메모리 결함을 탐지하기 위한 목적이라면 도 1에 도시된 바와 같이 정보 태그(2)에는 데이터 주소 공간(a), 데이터 초기화 여부(u), 및 주소 공간의 유효 사이즈(s)가 저장될 수 있다.

[0039] 본 발명에서 이러한 정보 태깅은 전술한 결함 탐지 모듈에 의해 수행되며 그러한 정보 태깅에 의해 생성된 태그 또한 결함 탐지 모듈에 의해서만 접근 가능하다. 구체적으로 설명하면, 어떤 모니터링 대상 모듈(예로써, malloc(), calloc() 등)이 대응되는 결함 탐지 모듈로 후킹되면, 프로세스 실행 과정에서는 결함 탐지 모듈이 대신 실행되며, 이때 결함 탐지 모듈은 그 실행에 관련된 데이터가 생성될 때 원본 데이터(1)만을 생성하지 않고 원본 데이터(1)에 정보 태그(2)가 태깅된 확장 데이터(3)가 생성되도록 한다. 그리고, 결함 탐지 모듈은 그 확장 데이터(3)가 사용되는 시점에서 확장 데이터(3)의 정보 태그(2)에 저장된 결함 탐지용 정보로부터 런-타임 결함 발생 가능성을 예측한다. 이처럼 결함 탐지 모듈은 정보 태깅 기능 및 결함 탐지 기능을 추가로 가짐으로써 일반적인 프로세스 모듈과 달리 런-타임 결함을 탐지할 수 있다.

- [0040] 모니터링 시점 후킹 및 런-타임 정보 태깅에 대한 이상의 설명에 기초하여, 이하에서는 본 발명에 따른 런-타임 결함 탐지 프로그램 및 런-타임 결함 탐지 방법의 실시예에 대해 설명한다.
- [0041] 먼저, 도 2를 참조하여 본 발명에 따른 런-타임 결함 탐지 프로그램의 일 실시예를 설명한다.
- [0042] 도 2를 참조하면, 본 발명의 실시예에 따른 런-타임 결함 탐지 프로그램(100)은, 컴포넌트 관리 모듈(110), 후킹 모듈(120), 결함 탐지 모듈들(130), 및 분석 모듈(140)을 포함한다.
- [0043] 컴포넌트 관리 모듈(110)은 다른 컴포넌트들(120, 130, 140)을 초기화하고 테스트 대상 프로세스에 호출하는 역할을 수행한다.
- [0044] 후킹 모듈(120)은 커널 공간에 접근 가능한 커널 드라이버로서 모니터링 대상 모듈들을 그에 대응하는 결함 탐지 모듈들(130)로 후킹하는 역할을 수행한다. 앞서 설명한 바와 같이, 후킹 모듈(120)에 의한 모니터링 시점 후킹은 프로세스 제어 블록(PCB)에 저장된 모니터링 대상 모듈의 주소를 결함 탐지 모듈의 주소로 후킹하는 방식으로 달성될 수 있다.
- [0045] 결함 탐지 모듈(130)은 정보 태깅 기능 및 결함 탐지 기능이 추가된 대체 모듈들로서, 런-타임 결함 탐지 프로그램(100)이 수행될 때, 대응되는 모니터링 대상 모듈들을 대신하여 실행된다.
- [0046] 분석 모듈(140)은 테스트 결과를 분석하는 모듈로서 결함 탐지 모듈(130)에서 수집하여 기록한 결함 유형 및 디버깅 정보가 저장된 테스트 로그를 분석하여 사용자 또는 개발자에게 결함의 위치와 원인, 디버깅 유효 정보들을 표시 장치 등을 통해 보고한다.
- [0047] 이러한 분석 모듈(140)은 테스트 대상 시스템에서 실행될 수도 있고, 테스트 대상 시스템과 유선 또는 무선으로 통신 가능한 별도의 호스트 기기에서 실행될 수도 있다. 분석 모듈(140)이 별도의 호스트 기기에서 실행되는 경우, 런-타임 결함 탐지 프로그램(100)은 분석 모듈(140)을 제외하고 전술한 모듈들(110, 120, 130)만으로 구성될 수도 있다.
- [0048] 이러한 런-타임 결함 탐지 프로그램(100)은 디지털 처리 장치에 의해 판독 가능한 기록매체에 저장되어 사용될 수 있다. 여기서 기록매체는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 디스크 등과 같은 디지털 처리 장치에 의해 판독 가능한 모든 기록 수단을 가리키며, 네트워크로 연결된 둘 이상의 컴퓨터 시스템에 의해 분산 방식으로 실행 가능한 유형의 것들까지 포함한다.
- [0049] 다음으로, 도 3 및 4를 참조하여 전술한 런-타임 결함 탐지 프로그램(100)을 이용한 런-타임 결함 탐지 방법의 일 실시예를 설명한다. 도 3은 본 발명의 실시예에 따른 런-타임 결함 탐지 방법을 나타낸 흐름도이며, 도 4는 런-타임 결함 탐지 방법의 제4 단계(결함 탐지 단계)를 구체적으로 보이는 흐름도이다.
- [0050] 도 3에 도시된 런-타임 결함 탐지 방법이 개시되면, S10 단계에서, 컴포넌트 관리 모듈(110)을 통해 테스트 대상 시스템의 커널에서 후킹 모듈(120)을 구동한다.
- [0051] 다음으로 S20 단계에서, 컴포넌트 관리 모듈(110)을 통해 런-타임 결함 테스트를 수행하고자 하는 프로세스들의 메모리 공간에 결함 탐지 모듈들(130)을 로딩시킨다. 전술한 바와 같이, 결함 탐지 모듈들(130)은 대응하는 모니터링 대상 모듈들의 기능에 더하여 정보 태깅 기능 및 결함 탐지 기능을 추가적으로 갖고 있다.
- [0052] 다음으로 S30 단계에서, 후킹 모듈(120)을 통해 모니터링 대상 모듈들의 실행 경로를 결함 탐지 모듈들(130)로 후킹한다. 이러한 모듈 후킹 단계가 수행되면, 모니터링 대상 모듈들의 실행 시점에서 그에 대응되는 결함 탐지 모듈들(130)이 대신 실행된다.
- [0053] 다음으로 S40 단계에서, 결함 탐지 모듈들(130)을 통해, 생성 데이터에 정보 태깅을 수행하고 데이터 사용 시점에 태그 정보를 검사하여 런-타임 결함을 탐지한다. 도 4를 참조하면, S40 단계는, 생성 데이터에 정보 태깅을 수행하는 단계(S41), 데이터 사용시 태그 정보를 검사하여 런-타임 결함을 탐지하는 단계(S42), 런-타임 결함 탐지에 의해 수집된 결함 유형 및 디버깅 정보를 테스트 로그에 기록하는 단계(S43), 및 모니터링 대상 모듈의 본래 기능을 수행하는 단계(S44)를 포함할 수 있다.
- [0054] 마지막으로 S50 단계에서, 분석 모듈(140)을 통해 테스트 로그에 기록된 결함 유형 및 디버깅 정보를 분석하여, 크래쉬 포인트와 디버깅 포인트를 구분하여 코드 위치 정보를 사용자에게 제공하고 결함 발생 시간, 결함이 발

생된 프로세스 명칭, 결함 원인, 결함 판단에 사용된 디버깅 정보를 함께 제공한다. 분석 모듈(140)에 의한 정보 제공은 디스플레이 장치를 통해 수행될 수 있으며, 그러한 디스플레이 장치는 모니터링 대상 시스템에 통합된 것일 수도 있고 별도의 호스트 기기에 구비된 것일 수도 있다.

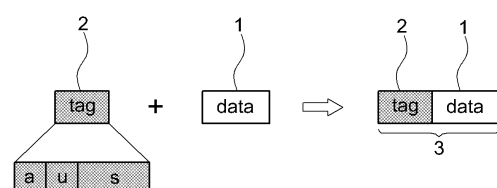
- [0055] 이상 설명한 바와 같이, 본 발명에 따른 런-타임 결함 탐지 방법에 의하면, 모니터링하고자 하는 대상 모듈들을 결함 탐지 모듈들로 후킹하고, 결함 탐지 모듈들의 정보 태깅 및 결함 탐지 기능을 통해 프로세스가 실행되는 도중 결함 유형과 디버깅 정보를 탐지한다.
- [0056] 이와 같이, 본 발명에 따른 런-타임 결함 탐지 방법은 결함 발생 이후에 히스토리 재현을 통해 결함을 분석하지 않고, 정보 태깅 기법을 사용하여 프로세스가 실행되는 런-타임에 결함 발생을 탐지하므로 디버깅이 필요한 코드 위치 정보 탐색의 정확성을 크게 개선할 수 있을 뿐만 아니라 그 위치 정보 탐색에 드는 노력을 최소화할 수 있다.
- [0057] 본 발명 이전에, 디버깅을 위한 하나의 방안으로서 프로세스가 런-타임에 사용한 메모리 정보를 별도의 메모리 공간에 기억하도록 하는 '쉐도우 메모리(Shadow memory)'라는 기법이 사용되었다. 하지만 쉐도우 메모리 기법은 프로세스 실행시 사용된 모든 메모리 정보를 별도의 주소에 저장하는 방식이어서 공간 추가 오버헤드가 많이 발생할 뿐만 아니라 성능 지연 오버헤드도 크게 발생한다.
- [0058] 반면, 본 발명에 따른 런-타임 결함 탐지 방법에 의하면, 정보 태깅을 통해 런-타임에 결함 발생 가능성을 판단하고 그 판단의 결과로 탐지되는 결함들만을 선별적으로 디버깅 정보로 제공하는 방식을 취하므로, 전술한 쉐도우 메모리 기법에서 발생되는 공간 추가 오버헤드 및 성능 지연 오버헤드 문제가 해소될 수 있다.
- [0059] 그리고, 본 발명에 따른 런-타임 결함 탐지 방법에 의하면, 프로세스가 실행되는 런-타임에 결함을 탐지하므로, 시스템이 다운되는 경우에 결함 탐지가 가능함은 물론, 테드락 상태 또는 메모리 누수와 같이 결함 증상이 겉으로 드러나지 않는 잠재 결함들에 대해서도 결함 탐지가 가능한 이점이 있다.
- [0060] 또한, 본 발명에 따른 런-타임 결함 탐지 방법은, 결함을 재현하는 방식을 취하지 않으므로, 사용자 입력 외에 다양한 외부 입력이 수신되는 차량용 네비게이션 시스템과 같은 임베디드 시스템에 특히 유용하게 적용될 수 있다.

## 부호의 설명

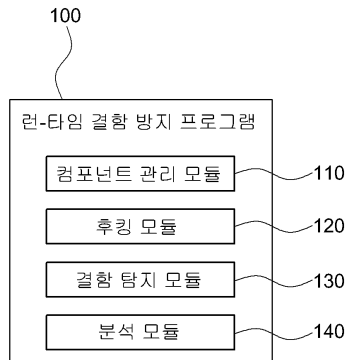
- [0061] 2 : 정보 태그  
3 : 확장 데이터  
100 : 런-타임 결함 방지 프로그램  
110 : 컴포넌트 관리 모듈  
120 : 후킹 모듈  
130 : 결함 탐지 모듈  
140 : 분석 모듈

## 도면

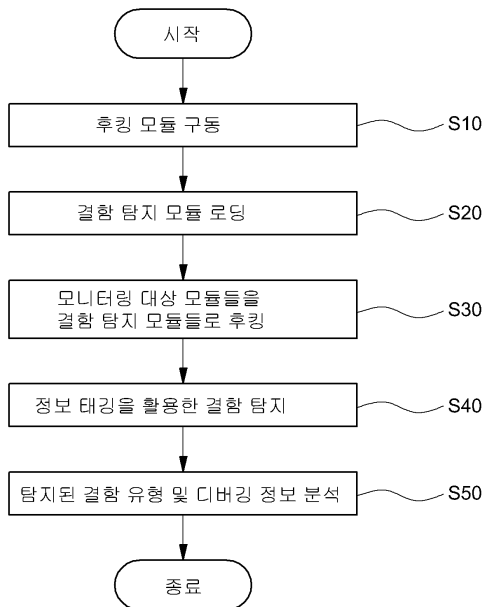
### 도면1



도면2



도면3



도면4

S40

