



(19) 대한민국특허청(KR)

(12) 등록특허공보(B1)

(45) 공고일자 2015년04월20일

(11) 등록번호 10-1513662

(24) 등록일자 2015년04월14일

(51) 국제특허분류(Int. Cl.)

G06F 11/36 (2006.01) G06F 3/048 (2006.01)

(21) 출원번호 10-2014-0135003

(22) 출원일자 2014년10월07일

심사청구일자 2014년10월07일

(56) 선행기술조사문헌

KR1020000031939 A*

KR1020060004921 A*

KR1020140033580 A*

*는 심사관에 의하여 인용된 문헌

(73) 특허권자

충남대학교산학협력단

대전광역시 유성구 대학로 99 (궁동, 충남대학교)

(72) 발명자

김형신

대전광역시 유성구 노은로426번길 183 (하기동)

이제민

대전광역시 유성구 은구비로 18, 1004호 (지족동, 가나파로스빌1)

한동건

충청남도 서산시 향교1로 12, 2동 906호 (동문동, 대림아파트)

(74) 대리인

특허법인 웰

전체 청구항 수 : 총 11 항

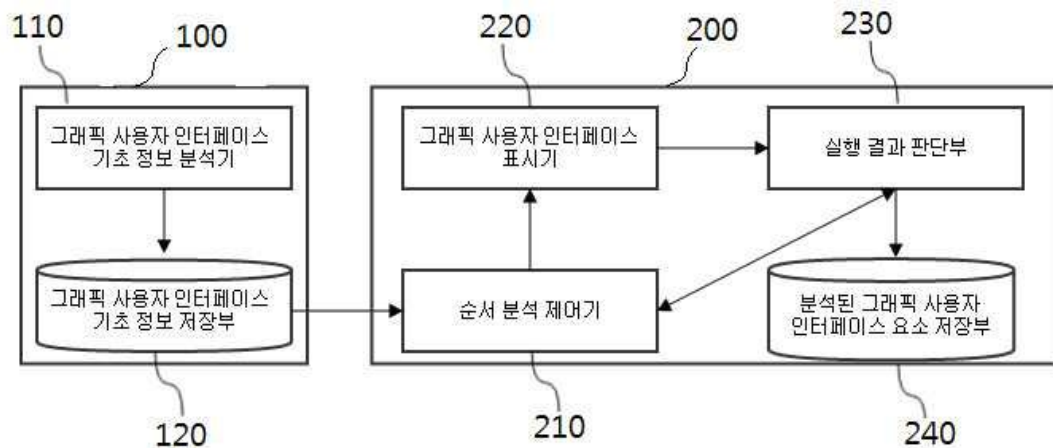
심사관 : 정성용

(54) 발명의 명칭 실행 가능한 그래픽 사용자 인터페이스 검색 시스템 및 검색 방법

(57) 요약

순서 의존성 없이 실행 가능한 그래픽 사용자 인터페이스(Graphic User Interface : GUI)를 찾는 시스템 및 방법에 관한 것으로, 소스코드 또는 역컴파일 기술을 이용해서 직접적인 실행 없이 그래픽 사용자 인터페이스에 대한 기초 정보를 분석하는 정적 분석 수단과 상기 정적 분석 수단에서 분석된 기초 정보에 따라 순차적으로 각각의 요소들을 실행하는 동적 분석 수단을 포함하는 구성을 마련하여, 정적과 동적 분석을 동시에 사용하므로 잘못된 결과를 발생시키지 않으며, 자동성을 높이게 될 수 있다.

대표도 - 도3



이 발명을 지원한 국가연구개발사업

과제고유번호 2014-0967-01

부처명 교육부

연구관리전문기관 충남대학교 산학협력단

연구사업명 2014년 충남대학교 자체연구과제 지원 사업

연구과제명 모바일기기 배터리 충방전 사이클 최적화

기 여 율 1/1

주관기관 충남대학교 컴퓨터공학과 임베디드 시스템 연구실

연구기간 2014.06.01 ~ 2015.05.31

명세서

청구범위

청구항 1

순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스를 찾는 시스템으로서,

소스코드 또는 역컴파일 기술을 이용해서 직접적인 실행 없이 그래픽 사용자 인터페이스에 대한 기초 정보를 분석하는 정적 분석 수단과

상기 정적 분석 수단에서 분석된 기초 정보에 따라 순차적으로 각각의 요소들을 실행하는 동적 분석 수단을 포함하고,

상기 동적 분석 수단은 상기 기초 정보 저장부로부터 기초정보를 읽어와 순차적으로 그래픽 사용자 인터페이스를 표시하는 작업을 생성하고 이를 분석하는 순서 분석 제어기, 상기 순서 분석 제어기에 의해서 할당된 작업을 담당하여 실제적인 그래픽 사용자 인터페이스를 생성하여 표시하는 그래픽 사용자 인터페이스 표시기 및 분석된 그래픽 사용자 인터페이스 요소를 저장하는 사용자 인터페이스 요소 저장부를 포함하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 2

제1항에 있어서,

상기 정적 분석 수단은 그래픽 사용자 인터페이스 기초 정보를 읽어내는 기초 정보 분석기와 상기 기초 정보를 저장하는 기초 정보 저장부를 포함하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 3

제2항에 있어서,

상기 동적 분석 수단은 상기 그래픽 사용자 인터페이스 표시기에 의해서 전자기기 상에 실제 표현된 정보가 정상적으로 출력되는지를 판단하는 실행결과 판단부를 더 포함하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 4

제2항에 있어서,

상기 기초 정보 분석기는 소프트웨어 명세, 소스코드 및 실행코드로부터 그래픽 사용자 인터페이스를 나타내는 기본 요소들에 대한 정보를 분석하고,

상기 기초 정보 저장부는 상기 기초 정보 분석부와 연결되어, 분석된 그래픽 사용자 인터페이스 기초 정보를 저장하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 5

제2항에 있어서,

상기 순서 분석 제어기는 상기 기초 정보 저장부와 연결되어, 그래픽 사용자 인터페이스 기초 정보를 순차적으로 읽어와서 정보를 표시하고, 정상적으로 수행되면, 다음 그래픽 사용자 인터페이스 기초 정보를 이용해서 다시 표시 실행을 명령하고, 문제가 있다면 해당 작업을 강제 종료시키고 오류를 처리하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 6

제3항에 있어서,

상기 사용자 인터페이스 요소 저장부는 상기 실행결과 판단부와 연결되어, 분석된 그래픽 사용자 인터페이스 요

소를 저장하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 7

제3항에 있어서,

상기 기초 정보는 버튼, 텍스트 박스의 그래픽 사용자 인터페이스 요소들의 종류와 크기, 배치, 색 중의 적어도 어느 하나인 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 8

제3항에 있어서,

상기 실행결과 판단부는 상기 그래픽 사용자 인터페이스 표시기에 정상적으로 표시가 되었다면 해당 그래픽 사용자 인터페이스 요소를 순서에 상관없이 실행 가능하다고 판단하여 상기 사용자 인터페이스 요소 저장부로 보내고, 정상적인 표시가 아닌 경우, 상기 순서 분석 제어기로 정상적인 표시가 아닌 결과를 통지하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템.

청구항 9

순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스를 찾는 방법으로서,

(a) 소프트웨어 명세, 소스코드 및 실행코드로부터 그래픽 사용자 인터페이스를 나타내는 기본 요소들에 대한 정보를 기초 정보 분석기에서 분석하는 단계,

(b) 상기 단계 (a)에서 분석된 그래픽 사용자 인터페이스 기초 정보를 기초 정보 저장부에 저장하는 단계,

(c) 상기 단계 (b)에서 저장된 기초정보를 순서 분석 제어기에서 읽어와 순차적으로 그래픽 사용자 인터페이스를 표시하는 작업을 생성하고 이를 분석하는 단계,

(d) 상기 순서 분석 제어기에 의해서 할당된 작업을 담당하여 실제적인 그래픽 사용자 인터페이스를 생성하여 표시하는 단계,

(e) 상기 단계 (d)에 의해서 전자기기 상에 실제 표현된 정보가 정상적으로 출력되는지를 실행결과 판단부에서 판단하는 단계,

(f) 상기 단계 (e)에서 정상으로 판단되는 경우, 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스 요소를 사용자 인터페이스 요소 저장부에 저장하는 단계를 포함하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 방법.

청구항 10

제9항에 있어서,

상기 단계 (d)에서 정상이 아닌 것으로 판단되면, 강제 종료하고 상기 단계 (c)로 진행하여 상기 단계 (b)에서 저장된 다른 기초 정보에 대해 분석을 실행하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 방법.

청구항 11

제9항에 있어서,

상기 단계 (e)에서 정상으로 판단되지 않은 경우, 순서 분석 제어기로 통지하고, 상기 단계 (c)로 진행하여 상기 단계 (b)에서 저장된 다른 기초 정보에 대해 분석을 실행하는 것을 특징으로 하는 실행 가능한 그래픽 사용자 인터페이스 검색 방법.

발명의 설명

기술 분야

본 발명은 동적으로 순서 의존성 없이 실행 가능한 그래픽 사용자 인터페이스(Graphic User Interface : GUI)를

찾는 시스템 및 방법에 관한 것으로, 특히 정적과 동적 분석을 동시에 사용해서 그래픽 사용자 인터페이스 간의 순서 의존성을 파악할 수 있는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템 및 검색 방법에 관한 것이다.

배경 기술

- [0002] IT 분야에 급속한 기술발전으로 인해 이에 따르는 소프트웨어의 기술 역시 발전되어야만 한다. 이와 같이, 소프트웨어는 컴퓨터 분야 및 통신 분야에 중요한 요소로서 자리하고 있다. 이로 인해, 소프트웨어의 신뢰성은 시스템 운용의 신뢰성과 직결되어 이에 대한 품질관리가 요구되고 있다. 소프트웨어의 품질관리는 그 대표적인 예로서, 소프트웨어 테스트 기법(software testing)을 가장 많이 사용하고 있다.
- [0003] 소프트웨어 테스트는 정적 테스트(Static Testing)와 동적 테스트(Dynamic Testing)으로 나뉜다. 동적 테스트는 구현된 소프트웨어를 직접 수행시켜서 그 결과가 원하는 기대 값과 같은지 판단하는 형태의 소프트웨어 테스트이고, 정적 테스트는 소프트웨어를 실제로 수행하지 않고 테스트하는 형태의 소프트웨어 테스트이다.
- [0004] 정적 테스트는 일반적으로 형상 인스펙션(configuration inspection), 소스 코드 인스펙션(source code inspection), 컴파일-빌드 프로세스(compile-build processor) 등으로 수행된다.
- [0005] 형상 인스펙션은 설정 값(Configuration value)의 제약사항 위반 여부를 검사하는 것이다. 소프트웨어를 작성할 때, 몇몇 변수는 정확한 값의 입력을 미치는 경우가 있으며, 형상은 설정 변수들을 구체적인 값으로 결정한 것을 나타낸다. 프로그램을 작성할 때, 몇몇 변수나 상수 값을 설정함으로써 프로그램의 컴파일이나 프로그램 실행에 영향을 주도록 할 수 있다. 이렇게 설정된 변수나 상수값을 프로그램의 형상이라고 하는데, 형상이 의도한 대로 되었는지 검사하는 것이 형상 인스펙션이다.
- [0006] 소스 코드 인스펙션은 소스 코드의 오류를 점검하는 활동으로 함수 및 메소드(method)의 시그니처(signature)가 명세서대로 작성되었는지 확인하거나 컴파일되기 위한 프로그램 변수 혹은 헤더 파일들이 적절하게 존재하는지 검사하는 것이다.
- [0007] 컴파일-빌드 프로세스는 컴파일하는 동안에 발생하는 에러에 대해서 보고하고 조치하는 행위이다.
- [0008] 또한, 안드로이드를 기반으로 하는 스마트폰의 보급률이 상승하고 안드로이드 OS 활용범위가 사회 전반으로 넓어지며, 다양한 기능을 하는 안드로이드 어플리케이션이 하루가 다르게 출시되고 있다.
- [0009] 안드로이드 어플리케이션은 기존의 자바 어플리케이션과는 다른 구조를 가지고 있으며, 모바일 디바이스와 같이 제한된 하드웨어 환경에서 동작하도록 최적화되어 있다. 이러한 안드로이드는 테스트를 위하여 JUnit, Monkey, MonkeyRunner의 세가지를 기본적으로 제공하고 있다.
- [0010] 또한, 단순한 API를 테스트하는 것 외에 Instrumentation based testing 기법을 제공하고 있다. 사용자의 행위에 대한 내용을 코드로 표현하여 기능 테스트를 한다. Monkey는 안드로이드의 무작위 이벤트를 타겟에 보내 에러가 발생하는지 또는 어느 정도의 시간을 버티는지를 테스트하는 스트레스 테스트를 할 수 있다. MonkeyRunner는 액션 좌표값과 액션 이벤트를 Jython으로 스크립트 형태의 코드를 제작하여 기능 테스트를 할 수 있는 도구이다.
- [0011] 이러한 기술의 일 예가 하기 문헌 1 및 2 등에 개시되어 있다.
- [0012] 예를 들어, 하기 특허문헌 1에는 도 1에 도시된 바와 같이, 차량용 소프트웨어 시스템의 기본 소프트웨어 모듈(11) 및 실행 환경 모듈(12)로부터 수신된 복수 개의 형상 소스 파일들(configuration source files)에 대한 적합성 검사를 수행하며, 전처리부(pre-processor)(21), 테스트 케이스 저장부(22), 테스트 제어부(23), 만족가능성 모듈로 이론 처리기(Satisfiability Modulo Theory solver : SMT)(24) 및 코드 실행부(25)를 포함하고, 기본 소프트웨어 모듈(11) 또는 실행 환경 모듈(12)을 구성하는 파일들 중 형상 파일에 대해 소프트웨어 정적 테스트를 수행하는 소프트웨어 정적 테스트 장치(20)에 대해 개시되어 있다.
- [0013] 또 하기 특허문헌 2에는 안드로이드 어플리케이션의 GUI 테스트를 위한 시나리오를 작성하기 이전의 단계인, 안드로이드 어플리케이션의 액티비티 기반 상태를 생성하는 방법으로서, 어플리케이션의 소스에 대한 역공학이 수행되는 단계, 상기의 수행된 역공학의 결과에 대해서, 액티비티를 중심으로 상태를 생성하는 단계 및 상기

의 상태도에 상기 액티비티의 라이프 사이클을 추가하는 단계를 포함하고, 상기 액티비티의 라이프 사이클을 추가하는 단계는, 상기 각각의 액티비티에 대해 피니시(finish) 함수가 존재하는지 여부를 체크하고 이벤트 백을 추가하는 단계를 포함하는 안드로이드 어플리케이션의 액티비티 기반 상태도 생성방법에 대해 개시되어 있다.

선행기술문헌

특허문헌

- [0014] (특허문헌 0001) 대한민국 공개특허공보 제2012-0072133호(2012.07.03 공개)
(특허문헌 0002) 대한민국 등록특허공보 제10-1416065호(2014.07.01 등록)

발명의 내용

해결하려는 과제

- [0015] 앱 마켓을 통해서 공급되는 많은 수의 응용프로그램들은 유용한 기능들을 제공하지만, 검증 과정의 비효율성 때문에 GUI 버그들을 포함하는 것이 많다. 특히, 크래쉬(Crash)를 유발시키는 GUI 버그들은 응용프로그램을 저장 없이 갑작스럽게 중단 시키므로, 매우 치명적이다. 모바일 테스트 연구는 많이 있지만, 기존의 연구들은 소스코드에 대한 의존도를 가지며 효율성과 자동성 떨어져서 앱 마켓에 있는 광범위한 응용프로그램에 대해서 GUI 테스트를 수행하기에는 한계가 존재한다.
- [0016] 즉, 상술한 바와 같은 종래의 기술에서는 정적으로 소스코드를 분석하거나 역컴파일 기술을 통해서 그래픽 유저 인터페이스들 간의 관계를 파악하므로, 직접 실행을 하는 방법이 아니기 때문에 많은 잘못된 결과를 가져오는 문제가 있었다.
- [0017] 또 상기와 같은 종래의 기술에서는 자동성을 떨어트려 최종적으로 많은 사람의 노력이 필요하다는 문제점도 있었다.
- [0018]
- [0019] 본 발명의 목적은 상술한 바와 같은 문제점을 해결하기 위해 이루어진 것으로서, 자동으로 소프트웨어를 구성하는 그래픽 사용자 인터페이스 요소들 간의 의존성을 파악해서, 독립적으로 실행 가능한 것을 찾아낼 수 있는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템 및 검색 방법을 제공하는 것이다.
- [0020] 본 발명의 다른 목적은 독립적으로 실행 가능한 그래픽 사용자 인터페이스들은 소프트웨어 테스트에 사용되어 높은 커버리지를 달성하게 할 수 있는 실행 가능한 그래픽 사용자 인터페이스 검색 시스템 및 검색 방법을 제공하는 것이다.

과제의 해결 수단

- [0021] 상기 목적을 달성하기 위해 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템은 는 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스를 찾는 시스템으로서, 소스코드 또는 역컴파일 기술을 이용해서 직접적인 실행 없이 그래픽 사용자 인터페이스에 대한 기초 정보를 분석하는 정적 분석 수단과 상기 정적 분석 수단에서 분석된 기초 정보에 따라 순차적으로 각각의 요소들을 실행하는 동적 분석 수단을 포함하는 것을 특징으로 한다.
- [0022] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템에 있어서, 상기 정적 분석 수단은 그래픽 사용자 인터페이스 기초 정보를 읽어내는 기초 정보 분석기와 상기 기초 정보를 저장하는 기초 정보 저장부를 포함하고, 상기 동적 분석 수단은 상기 기초 정보 저장부로부터 기초정보를 읽어와 순차적으로 그래픽 사용자 인터페이스를 표시하는 작업을 생성하고 이를 분석하는 순서 분석 제어기, 상기 순서 분석 제어기에 의해서 할당된 작업을 담당하여 실제적인 그래픽 사용자 인터페이스를 생성하여 표시하는 그래픽 사용자 인터페이스 표시기 및 분석된 그래픽 사용자 인터페이스 요소를 저장하는 사용자 인터페이스 요소 저장부를 포함하는 것을 특

정으로 한다.

- [0023] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템에 있어서, 상기 동적 분석 수단은 상기 그래픽 사용자 인터페이스 표시기에 의해서 전자기기 상에 실제 표현된 정보가 정상적으로 출력되는지를 판단하는 실행결과 판단부를 더 포함하는 것을 특징으로 한다.
- [0024] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템에 있어서, 상기 기초 정보 분석기는 소프트웨어 명세, 소스코드 및 실행코드로부터 그래픽 사용자 인터페이스를 나타내는 기본 요소들에 대한 정보를 분석하고, 상기 기초 정보 저장부는 상기 기초 정보 분석부와 연결되어, 분석된 그래픽 사용자 인터페이스 기초 정보를 저장하는 것을 특징으로 한다.
- [0025] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템에 있어서, 상기 순서 분석 제어기는 상기 기초 정보 저장부와 연결되어, 그래픽 사용자 인터페이스 기초 정보를 순차적으로 읽어와서 정보를 표시하고, 정상적으로 수행되면, 다음 그래픽 사용자 인터페이스 기초 정보를 이용해서 다시 표시 실행을 명령하고, 문제가 있다면 해당 작업을 강제 종료시키고 오류를 처리하는 것을 특징으로 한다.
- [0026] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템에 있어서, 상기 사용자 인터페이스 요소 저장부는 상기 실행결과 판단부와 연결되어, 분석된 그래픽 사용자 인터페이스 요소를 저장하는 것을 특징으로 한다.
- [0027] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템에 있어서, 상기 기초 정보는 버튼, 텍스트 박스의 그래픽 사용자 인터페이스 요소들의 종류와 크기, 배치, 색 등의 적어도 어느 하나인 것을 특징으로 한다.
- [0028] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템에 있어서, 상기 실행결과 판단부는 상기 그래픽 사용자 인터페이스 표시기에 정상적으로 표시가 되었다면 해당 그래픽 사용자 인터페이스 요소를 순서에 상관없이 실행 가능하다고 판단하여 상기 사용자 인터페이스 요소 저장부로 보내고, 정상적인 표시가 아닌 경우, 상기 순서 분석 제어기로 정상적인 표시가 아닌 결과를 통지하는 것을 특징으로 한다.
- [0029] 또한, 상기 목적을 달성하기 위해 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 방법은 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스를 찾는 방법으로서, (a) 소프트웨어 명세, 소스코드 및 실행코드로부터 그래픽 사용자 인터페이스를 나타내는 기본 요소들에 대한 정보를 기초 정보 분석기에서 분석하는 단계, (b) 상기 단계 (a)에서 분석된 그래픽 사용자 인터페이스 기초 정보를 기초 정보 저장부에 저장하는 단계, (c) 상기 단계 (b)에서 저장된 기초정보를 순서 분석 제어기에서 읽어와 순차적으로 그래픽 사용자 인터페이스를 표시하는 작업을 생성하고 이를 분석하는 단계, (d) 상기 순서 분석 제어기에 의해서 할당된 작업을 담당하여 실제적인 그래픽 사용자 인터페이스를 생성하여 표시하는 단계, (e) 상기 단계 (d)에 의해서 전자기기 상에 실제 표현된 정보가 정상적으로 출력되는지를 실행결과 판단부에서 판단하는 단계, (f) 상기 단계 (e)에서 정상으로 판단되는 경우, 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스 요소를 사용자 인터페이스 요소 저장부에 저장하는 단계를 포함하는 것을 특징으로 한다.
- [0030] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 방법에 있어서, 상기 단계 (d)에서 정상이 아닌 것으로 판단되면, 강제 종료하고 상기 단계 (c)로 진행하여 상기 단계 (b)에서 저장된 다른 기초 정보에 대해 분석을 실행하는 것을 특징으로 한다.
- [0031] 또 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 방법에 있어서, 상기 단계 (e)에서 정상으로 판단되지 않은 경우, 순서 분석 제어기로 통지하고, 상기 단계 (c)로 진행하여 상기 단계 (b)에서 저장된 다른 기초 정보에 대해 분석을 실행하는 것을 특징으로 한다.

발명의 효과

- [0032] 상술한 바와 같이, 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템 및 검색 방법에 의하면, 정적과 동적 분석을 동시에 사용하므로 잘못된 결과를 발생시키지 않으며, 자동성을 높일 수 있다는 효과가 얻어진다.
- [0033] 즉, 본 발명에 따르면, 기존 방법들과 다르게 그래픽 사용자 인터페이스들 간의 순서 의존성을 파악하기 위해서 동적 방법을 이용하므로, 직접적인 실행을 이용한 동적 방법을 이용할 경우 잘못된 결과를 만들어 내지 않으며,

사람의 개입을 필요로 하지 않은 완벽한 자동성을 제공할 수 있다.

- [0034] 또, 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템 및 검색 방법에 의하면, 분석한 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스 요소들은 소프트웨어 테스트에 활용되어 더 높은 커버리지를 달성할 수 있다는 효과도 얻어진다.

도면의 간단한 설명

- [0035] 도 1은 종래의 소프트웨어 정적 테스트 장치의 구성을 도시한 도면,
 도 2는 본 발명에 적용되는 자동화된 블랙박스 테스트 구조를 나타내는 도면,
 도 3은 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템의 구성도,
 도 4는 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 방법을 설명하기 위한 흐름도.

발명을 실시하기 위한 구체적인 내용

- [0036] 본 발명의 상기 및 그 밖의 목적과 새로운 특징은 본 명세서의 기술 및 첨부 도면에 의해 더욱 명확하게 될 것이다.
- [0037] 먼저 본 발명에 따른 특징에 대해 기술한다.
- [0038] 본 발명은 소프트웨어 검증 분야에 활용될 수 있는 것으로서, 특히 구체적인 내부 정보 없이 검증해야 하는 블랙박스 검증 기법에 활용되어 높은 커버리지를 달성할 수 있다. 그래픽 사용자 인터페이스 간의 순서 의존성을 파악해서 의존성이 없는 독립적인 그래픽 사용자 인터페이스가 발견되면, 이러한 정보는 각각의 테스트 시작점으로 활용될 수 있다. 기존의 단일 시작 지점의 그래픽 사용자 인터페이스를 활용한 블랙박스 테스트와 다르게 분석된 다수의 의존성이 없는 그래픽 사용자 인터페이스를 이용할 경우 더 높은 커버리지를 달성할 수 있다.
- [0039] 또한, 기존의 기술은 사람의 개입을 필요로 한다. 따라서 현재 급증하고 있는 앱 마켓 시장을 포용할 수 없다. 현재 앱 마켓에 등록된 소프트웨어의 수는 100만개 이상이며, 이러한 소프트웨어를 모두 검증하기 위해서는 높은 커버리지를 보장하며 자동화를 지원하는 기술이 필요하다. 본 발명은 이러한 두 가지 측면을 모두 충족하기 때문에 현재의 모바일 시장에 매우 적합하다.
- [0040] 기존 소프트웨어 테스트 시작은 크게 화이트 박스와 블랙박스 부분으로 나누어진다. 본 발명은 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스를 자동으로 분석하기 때문에 블랙박스 테스트 분야에 적극적으로 사용될 수 있다. 또 소프트웨어의 신뢰성을 보장하기 위해서는 테스트 기술이 필수적이며, 현재의 앱 스토어 같이 엄청난 테스트를 필요로 하는 시장을 고려해 볼 때 자동 테스트는 매우 중요한 기술이다. 따라서 본 발명은 자동성을 제공하며 블랙박스 테스트에 활용될 수 있다는 점에서 매우 경제성이 높은 기술이다.
- [0041] 다음에, 도 2에 따라 본 발명에 사용되는 블랙박스 테스트 구조에 대해 기술한다. 도 2는 본 발명에 적용되는 자동화된 블랙박스 테스트 구조를 나타내는 도면이다.
- [0042] 도 2에 도시된 바와 같이, 본 발명에서 제안한 방법은 호스트 부분과 타겟 두 부분으로 나뉘어진다. 호스트 부분은 자동 테스트를 위해서 설치파일 리버싱, 액티비티 순서 분석, 이벤트 발생, 로거 모듈들로 구성된다. 각각의 모듈들은 파이썬(Python)으로 개발되었다. 타겟 부분은 자동 테스트를 위해서 사용한 안드로이드 컴포넌트들을 나타낸다. 해당 안드로이드 컴포넌트들은 자동 테스트를 위해서 약간 수정하였다. 수정을 위해서 시아노젠 모드(Cyanogen-mod-7)의 넥서스원(Nexus-one)버전을 사용했다.
- [0043] 전체적인 자동 테스트 진행 과정은 다음과 같다. 안드로이드 설치파일인(.apk) 파일로부터 액티비티 정보들을 추출한 다음, 자동으로 각각의 액티비티들에 대한 순서 의존성을 파악한다. 그 후에, 순서에 상관없이 실행 가능한 각각의 액티비티들에 대해서 무작위 이벤트를 이용한 GUI 테스트를 수행하고 실시간으로 테스트 결과를 기록한다. 마지막으로 테스트 결과를 요약해서 보여주게 된다. 각 모듈에 대한 상세한 설명과 구현 내용은 다음과 같다.
- [0044] 설치파일 리버싱 모듈: 응용프로그램의 설치파일에서 중요한 내부 정보를 담고 있는 매니페스트(Manifest.xml) 파일을 역패키징 기술을 이용해서 얻어낸다. 이 파일을 파싱해서 응용프로그램의 패키지 이름, 버전, 액티비티, 퍼미션 정보 등을 알아내며, 응용프로그램을 타겟에 설치한다.

- [0045] 액티비티 순서 분석 모듈: 안드로이드 GUI를 구성하는 기본단위는 액티비티이며, 응용프로그램은 메인 액티비티로부터 시작해서 이벤트에 따라서 다른 종류의 액티비티로 넘어가게 된다. 응용프로그램의 액티비티들은 액티비티 관리자(Activity Manager)에 의해서 등록되고 관리된다. 해당 모듈에서는 응용프로그램의 액티비티 정보와 액티비티 관리자를 이용해서 순서에 상관없이 실행 가능한 액티비티들을 자동으로 분석하는 작업을 하며, 방법은 다음과 같다. 액티비티 실행을 담당할 새로운 프로세스를 생성한다. 생성한 프로세스는 응용프로그램에 정의된 액티비티들 중 분석되지 않은 것을 한 개 가져온다. 그 다음 액티비티 관리자에게 그것을 실행하게 하는 명령을 내린다. 액티비티 관리자는 의존성이 없는 액티비티의 경우 정상적으로 실행결과를 호스트로 응답하게 된다. 이 경우에는 순서에 상관없이 실행 가능한 액티비티로 판단하고 결과를 저장한다. 하지만, 데이터베이스 정보와 같은 특정 파라미터를 필요로 해서, 순서에 의존적인 액티비티일 경우 결과가 반환되지 않으며, GUI 크래시가 발생 된다. 이 경우에는 액티비티 실행을 담당하는 생성한 프로세스를 시그널을 이용해서 종료시킨다. 그리고 액티비티 관리자를 이용해서 해당 응용프로그램을 강제 종료 시킨다. 의존성을 파악할 액티비티가 남아있으면, 위의 과정을 반복하게 된다.
- [0046] 이벤트 발생 모듈: 기존의 몽키의 테스트 시작은 항상 메인 액티비티였던 것과 다르게, 본 발명의 방법은 분석을 통해 알아낸 순서에 상관없이 실행 가능한 액티비티들에 대해서 무작위 이벤트 생성을 통한 GUI 테스트를 수행한다. 이를 위해서 안드로이드 플랫폼의 몽키 서버(Monkey Server) 부분을 수정하여, 지정된 액티비티에 대해서 테스트가 가능하도록 했다.
- [0047] 로거 모듈: 테스트의 모든 과정에 대한 결과를 각각의 응용프로그램에 맞추어서 파일에 저장 한다. 테스트 결과는 설치와 제거에 대한 성공 여부, 액티비티 커버리지 정보, GUI 크래쉬 발생 수와 상세한 에러 메시지 정보로 구성된다.
- [0048] 이하, 본 발명의 특징적인 구성을 도 3에 따라서 설명한다.
- [0049] 도 3은 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템의 구성도로서, 도 3은 그래픽 사용자 인터페이스 요소들 중에서 순서에 상관없이 실행 가능한 것들을 찾는 시스템을 나타낸다.
- [0050] 도 3에 도시된 바와 같이, 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템은 순서 없이 실행 가능한 그래픽 사용자 인터페이스를 찾는 시스템으로서, 소스코드 또는 역컴파일 기술을 이용해서 직접적인 실행 없이 그래픽 사용자 인터페이스에 대한 기초 정보를 분석하는 정적 분석 수단(100)과 상기 정적 분석 수단(100)에서 분석된 기초 정보에 따라 순차적으로 각각의 요소들을 실행하는 동적 분석 수단(200)을 포함한다.
- [0051] 상기 정적 분석 수단(100)과 동적 분석 수단(200)은 프로세서와 메모리를 구비한 PC, 노트북, 넷북, PDA, 모바일, 스마트폰에 의해 실행되거나 프로그램에 의해 실행될 수 있다.
- [0052] 상기 동적 분석 수단(200)은 상기 정적 분석 수단(100)에서 분석된 기초 정보를 이용해서 순차적으로 각각의 요소들을 실행하고, 실행 결과를 분석하여 해당 그래픽 사용자 인터페이스가 순서에 상관없이 실행 가능한지를 최종적으로 검색한다.
- [0053] 상기 정적 분석 수단(100)은 그래픽 사용자 인터페이스 기초 정보를 읽어내는 기초 정보 분석기(110)와 상기 기초 정보를 저장하는 기초 정보 저장부(120)를 포함한다.
- [0054] 상기 기초 정보 분석기(110)는 그래픽 사용자 인터페이스의 기초 정보 분석기로써 요소들의 종류, 크기, 배치, 색 등과 같은 실제 표현을 위한 정보를 알아낸다. 이러한 정보들은 소스 코드 또는 이진 코드로부터 알아낼 수 있다. 즉, 상기 기초 정보 분석기(110)는 소스코드나, 역컴파일 기술을 이용해서 직접적인 실행 없이 그래픽 사용자 인터페이스에 대한 기초 정보를 알아낸다. 구체적으로 버튼, 텍스트 박스 등과 같은 각각의 그래픽 사용자 인터페이스 요소들의 종류와 크기, 배치, 색 등을 추출하여 분석한다.
- [0055] 상기 기초 정보 저장부(120)는 상기 기초 정보 분석기(110)에 의해서 분석된 그래픽 사용자 인터페이스에 대한 정보를 저장하고, 저장된 정보를 이용해서 실제 그래픽 인터페이스 요소들을 그리며, 이를 동적 분석 수단(200)으로 전달한다.
- [0056] 상기 동적 분석 수단(200)은 상기 기초 정보 저장부(120)로부터 기초정보를 읽어와 순차적으로 그래픽 사용자 인터페이스를 표시하는 작업을 생성하고 이를 분석하는 순서 분석 제어기(210), 상기 순서 분석 제어기(210)에 의해서 할당된 작업을 담당하여 실제적인 그래픽 사용자 인터페이스를 생성하여 표시하는 그래픽 사용자 인터페이스 표시기(220), 상기 그래픽 사용자 인터페이스 표시기(220)에 의해서 전자기기 상에 실제 표현된 정보가 정

상적으로 출력되는지를 판단하는 실행결과 판단부(230) 및 분석된 그래픽 사용자 인터페이스 요소를 저장하는 사용자 인터페이스 요소 저장부(240)를 포함한다.

- [0057] 상기 순서 분석 제어기(210)는 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스를 분석하기 위해서 직접적인 실행을 이용한다. 이때 각각의 실행을 제어하는 기능을 담당한다. 상기 순서 분석 제어기(210)는 상기 기초 정보 저장부(120)로부터 순차적으로 그래픽 사용자 인터페이스의 기초정보를 읽어 와서 정보를 표시하는 기능을 담당하는 그래픽 사용자 인터페이스 표시기(220)로 해당 정보를 보내고 결과를 기다린다. 그래픽 사용자 인터페이스 표시기(220)에서의 결과에 따라 정상적으로 수행되었다면, 다음 그래픽 사용자 인터페이스 기초 정보를 이용해서 다시 표시 실행을 명령하고, 그래픽 사용자 인터페이스 표시기(220)에서 문제가 있다면, 해당 작업을 강제 종료시키고 오류를 처리하게 한다. 본 발명에서는 이와 같은 처리를 통해서 안정적으로 작업자의 개입 없이도 분석을 계속 할 수 있다.
- [0058] 상기 그래픽 사용자 인터페이스 표시기(220)는 기초 정보를 이용해서 실행을 통한 직접적인 표현을 수행한다. 이때 해당 그래픽 사용자 인터페이스가 특정 사전 데이터에 의존적이라면 표시도중 오류가 발생한다. 상기 그래픽 사용자 인터페이스 표시기(220)는 일반적인 그래픽 사용자 인터페이스를 표시하는 기능을 하며 사전 데이터 의존성 같은 어떠한 분석도 수행하지 않고, 단순히 기초정보를 이용해서 표시하는 기능을 담당한다.
- [0059] 상기 실행결과 판단부(230)는 상기 그래픽 사용자 인터페이스 표시기(220)에 의해서 표시된 그래픽 사용자 인터페이스가 정상적으로 표시되었는지를 판단한다. 즉, 오류 메시지, 콜 스택 정보 등을 이용해서 정상적으로 표시 가능한 그래픽 사용자 인터페이스 요소인지를 분석하고, 정상적으로 표시가 되었다면 해당 그래픽 사용자 인터페이스 요소는 순서에 상관없이 실행 가능하다고 판단하여 사용자 인터페이스 요소 저장부(240)로 보내고, 아니라면, 순서 분석 제어기(210)로 이러한 결과를 통지한다.
- [0060] 상기 사용자 인터페이스 요소 저장부(240)는 최종적으로 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스 요소를 저장하고, 이러한 정보는 다양한 소프트웨어 테스트 기법에 활용된다.
- [0061] 다음에 도 4에 따라 본 발명의 실행 가능한 그래픽 사용자 인터페이스 검색 방법을 설명한다.
- [0062] 도 4는 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 방법을 설명하기 위한 흐름도이다.
- [0063] 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색은 순서 없이 실행 가능한 그래픽 사용자 인터페이스를 찾는 방법으로서, 먼저 소프트웨어 명세, 소스코드 및 실행코드로부터 그래픽 사용자 인터페이스를 나타내는 기본 요소들에 대한 정보를 기초 정보 분석기(110)에서 분석한다(S10). 이와 같은 분석에 의해 요소들의 종류, 크기, 배치, 색 등과 같은 실제 표현을 위한 정보를 알아낼 수 있다.
- [0064] 상기 단계 S10에서 분석된 그래픽 사용자 인터페이스 기초 정보를 기초 정보 저장부(120)에 저장되고(S20), 순서 분석 제어기(210)로 전송된다.
- [0065] 상기 단계 S20에서 저장된 기초정보를 순서 분석 제어기(210)에서 읽어와 순차적으로 그래픽 사용자 인터페이스를 표시하는 작업을 생성하고 이를 분석하고(S30), 정상인기 아닌가를 판단한다(S40).
- [0066] 상기 단계 S40에서 정상인 것으로 판단되면, 상기 순서 분석 제어기(210)에 의해서 할당된 작업을 담당하여 실제적인 그래픽 사용자 인터페이스를 생성하여 그래픽 사용자 인터페이스 표시기(220)에 표시하게 한다(S50).
- [0067] 한편, 상기 단계 S40에서 정상이 아닌 것으로 판단되면, 강제 종료하고(S60), 상기 단계 S30으로 진행하여 상기 단계 S20에서 저장된 다른 기초 정보에 대해 계속하여 분석을 실행한다.
- [0068] 다음에 상기 단계 S50에 의해서 전자기기 상에 실제 표현된 정보가 정상적으로 출력되는지를 실행결과 판단부(230)에서 판단한다(S70).
- [0069] 상기 단계 S70에서 정상으로 판단되는 경우, 순서에 상관없이 실행 가능한 그래픽 사용자 인터페이스 요소를 사용자 인터페이스 요소 저장부(240)에 저장하고(S80), 상기 단계 S30으로 진행하여 상술한 과정을 반복한다.
- [0070] 한편, 상기 단계 S70에서 정상으로 판단되지 않은 경우, 순서 분석 제어기(210)로 통지하고, 상기 단계 S30으로 진행하여 상기 단계 S20에서 저장된 다른 기초 정보에 대해 분석을 계속 실행한다.
- [0071] 상술한 바와 같이 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 방법에 의한 실험 결과는 하기와 같다.

- [0072] 본 발명에서 자동성과 효율성을 검증하기 위해 수행한 두 가지 실험에 대해서 설명한다. 두 실험 모두 제안한 방법과 기존 몽키를 이용해서 수행한 다음 결과를 비교했다. 첫 번째는 오픈소스 응용프로그램인 워드프레스(WordPress)에 대해서 코드 커버리지를 측정하고 알려진 GUI 버그의 검출률을 보이는 실험이다. 두 번째는 앱마켓에서 선별한 20개에 대해서 액티비티 커버리지를 측정하고 GUI 크래시를 검출하는 실험이다.
- [0073] 먼저, 오픈소스 응용프로그램의 커버리지 측정을 실험하였다.
- [0074] 즉, 본 발명의 방법의 효율성을 검증하기 위해서 코드 커버리지와 알려진 버그들을 검출하는 실험을 했다. 코드 커버리지 측정을 위해서 오픈 소스 응용프로그램인 워드프레스를 이용했으며, 넥서스원에서 실험했다. 워드프레스는 블로그 활동 지원을 위한 응용으로, 4년간의 변경 이력을 확인할 수 있다. 선택한 버전은 깃 커밋 해시(Git commit hash) 66846425이다. 워드프레스에 두 개의 계정을 연동시키고 각각의 블로그에는 멀티미디어 데이터를 포함한 여러 포스트들을 추가했다. 코드 커버리지를 측정하기 위해서 추가로 안드로이드 인스트루멘테이션 클래스(Android Instrumentation Class)와 몽키 라이브러리 그리고 엠마(EMMA)를 이용했다.
- [0075] 실험을 위한 테스트 시나리오는 다음과 같다. 구현된 도구를 이용해서 무작위 이벤트 2,000개를 자동으로 분석한 독립적인 액티비티에 대해서 테스트 하도록 했다. 독립적인 액티비티는 14개가 있었고, 총 28,000개의 무작위 이벤트를 이용한 테스트가 수행되었다. 같은 조건으로 테스트 하기 위해서 기존 몽키를 이용한 테스트의 경우 2,000개의 무작위 이벤트를 이용해서 메인 액티비티를 시작으로 14번 테스트를 했다.
- [0076] 두 방법을 이용한 실험 결과, 본 발명에 따른 방법은 구문과 함수 커버리지를 각각 32%와 34%를 달성했으며, 액티비티 커버리지는 59.2%를 달성했다. 이러한 결과는 같은 조건에서 몽키를 이용해서 테스트 했을 때보다 구문 커버리지가 15% 높은 수치이다. 또한, 종래의 몽키를 이용해서 4.46시간 동안 45,000개의 무작위 이벤트로 테스트 했을 때 달성한 25.27%의 구문 커버리지 보다 높은 수치이었다.
- [0077] 다음에 마켓 응용프로그램들의 커버리지 측정을 실험하였다.
- [0078] 본 발명의 효율성을 실제 마켓 응용프로그램들에 대해서도 입증하기 위해서 실험을 했다. 실험에 사용한 20개의 응용프로그램들은 다양한 카테고리 and 평점 그리고 누적 다운로드를 가진다. 실험은 응용프로그램의 설치파일만을 가지고 수행했기 때문에, 블랙박스 수준의 결과만을 얻어냈다. 우선, 액티비티 커버리지에 대한 실험 결과는 본 발명에 따른 방법의 평균 액티비티 커버리지는 64%를 나타냈으며, 기존의 몽키의 경우 22%를 나타냈다. 퍼펙트 뷰어(PerfectViewer) 응용프로그램의 경우 본 발명의 방법으로 100% 액티비티 커버리지를 달성했다. 또한 몇몇 응용프로그램의 경우 기존 몽키보다 3배 이상 높은 액티비티 커버리지를 달성했다. 20개의 응용프로그램에 대해서 크래시를 분석한 결과, 본 발명의 방법의 경우 9개의 응용프로그램들에 대해서 크래시를 검출할 수 있었다. 반면에, 기존 몽키의 경우에는 4개의 응용프로그램들에 대해서만 크래시가 검출되었다. 또한 각각의 응용프로그램에 대해서 최대 4배 이상 많은 크래시를 검출하였다. 이러한 결과는, 크래시의 검출 면에서도 본 발명이 우수하다는 것을 보여 준다.
- [0079] 이상 본 발명자에 의해서 이루어진 발명을 상기 실시 예에 따라 구체적으로 설명하였지만, 본 발명은 상기 실시 예에 한정되는 것은 아니고 그 요지를 이탈하지 않는 범위에서 여러 가지로 변경 가능한 것은 물론이다.

산업상 이용가능성

- [0080] 본 발명에 따른 실행 가능한 그래픽 사용자 인터페이스 검색 시스템 및 검색 방법을 사용하는 것에 의해 기존의 몽키를 이용한 테스트 방법보다 높은 코드 커버리지와 GUI 버그의 검출을 나타내므로, 안드로이드 마켓에 있는 GUI 버그를 포함하는 응용프로그램들을 걸러낼 수 있을 것이다.

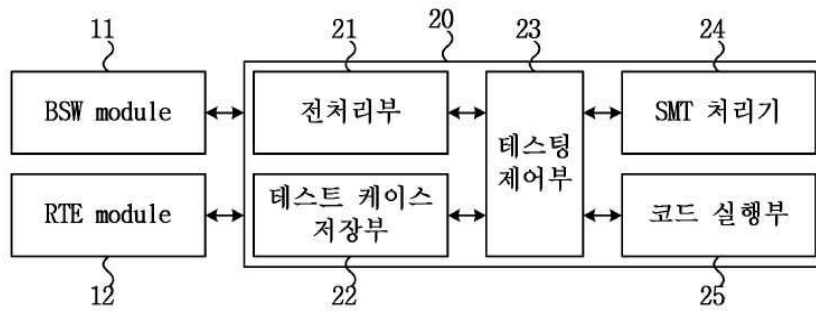
부호의 설명

- [0081] 100 : 정적 분석 수단
110 : 기초 정보 분석기
120 : 기초 정보 저장부

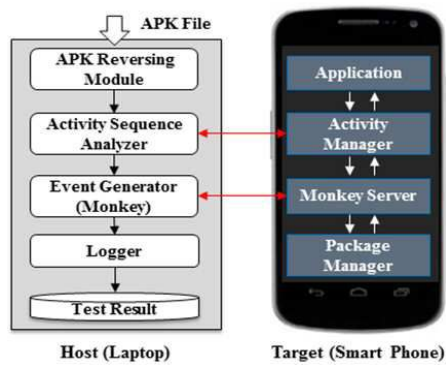
- 200 : 동적 분석 수단
- 210 : 순서 분석 제어기
- 220 : 그래픽 사용자 인터페이스 표시기
- 230 : 실행결과 판단부
- 240 : 사용자 인터페이스 요소 저장부

도면

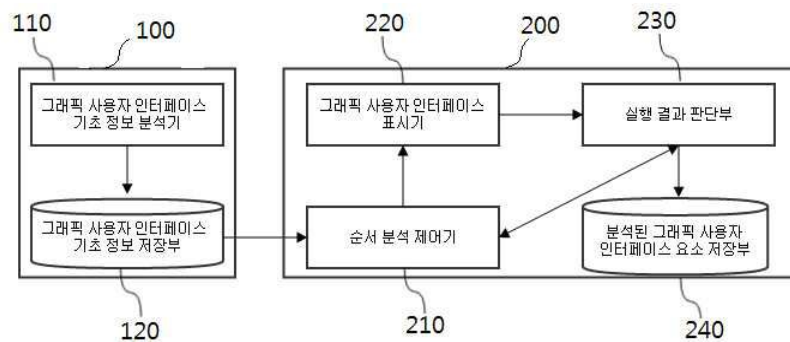
도면1



도면2



도면3



도면4

