(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년05월28일
(11) 등록번호 10-1266685
(24) 등록일자 2013년05월15일

(51) 국제특허분류(Int. Cl.)
G06F 9/44 (2006.01) *G06F 17/00* (2006.01)
(21) 출원번호 10-2010-7012154
(22) 출원일자(국제) 2008년11월03일
심사청구일자 2011년01월20일
(85) 번역문제출일자 2010년06월01일
(65) 공개번호 10-2010-0099690
(43) 공개일자 2010년09월13일
(86) 국제출원번호 PCT/US2008/082240
(87) 국제공개번호 WO 2009/059291
국제공개일자 2009년05월07일
(30) 우선권주장
11/982,675 2007년11월02일 미국(US)
(56) 선행기술조사문헌
US05826005 A

(73) 특허권자
티티아이 인벤션스 씨 엘엘씨
미국 델라웨어 윌밍톤 스위트 400 센터빌 로드 2711 (우: 19808)
(72) 발명자
코친왈라 퓨너
미국 07920 뉴저지주 배스킹 릿지 사우스 얼워드 애비뉴 154
미칼레프 조세핀
미국 07040 뉴저지주 매이플우드 미들랜드 불러바드 78
울러트 존 알 2세
미국 08836 뉴저지주 마틴스빌 매이플라워 코트 1106
(74) 대리인
특허법인코리아나

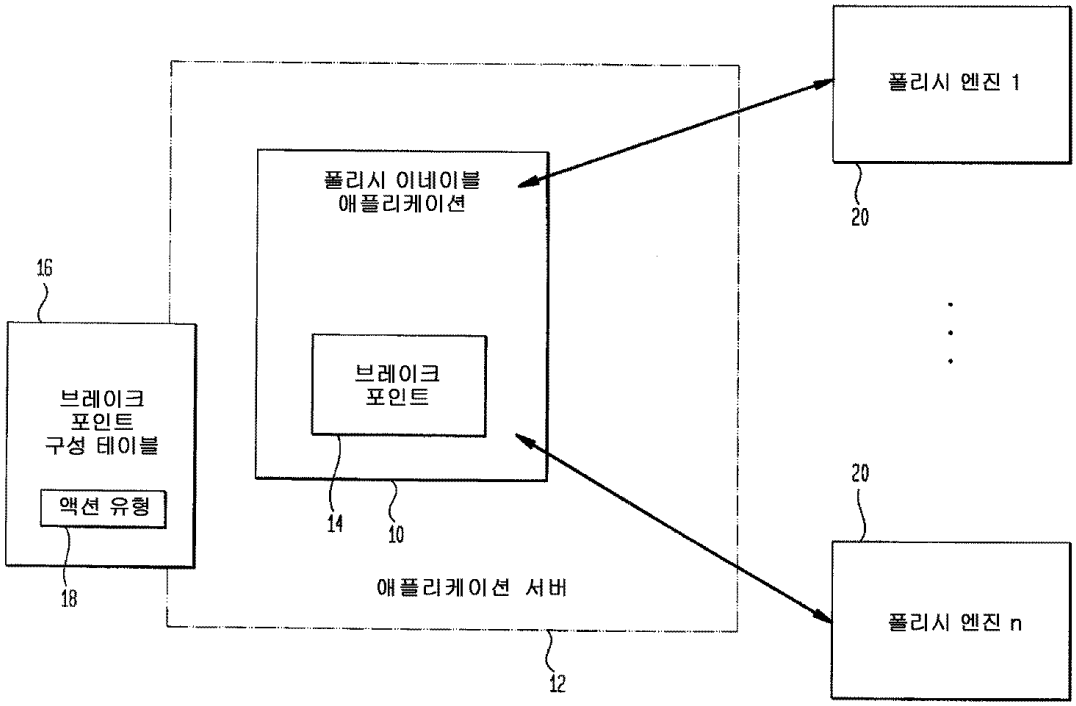
전체 청구항 수 : 총 13 항 심사관 : 김종기

(54) 발명의 명칭 **폴리시 이네이블 프로그래밍을 위한 방법 및 시스템**

(57) 요 약

애플리케이션 코드에 대한 변형 없이 통신 시스템에서 애플리케이션의 애플리케이션 거동의 외부 실행-시간 적응을 허용하기 위한 시스템 및 방법은, 적어도 하나의 브레이크 포인트를 갖고 애플리케이션 서버 상에 상주하는 애플리케이션, 적어도 하나의 식별 가능한 결정 엔진, 및 각각의 브레이크 포인트에 대해 결정 엔진의 적어도 하나의 식별자를 갖는 브레이크 포인트들의 리스트를 포함하여, 브레이크 포인트들 중 하나에서, 애플리케이션은 브레이크 포인트들의 리스트를 액세스하고, 브레이크 포인트에 대응하는 결정 엔진의 리스팅된 인스턴스들을 인보크하며, 결정 엔진에 기초하여 애플리케이션 거동을 적응시키고, 또한, 브레이크 포인트들의 리스트 내의 각 엔트리는 시퀀스 넘버를 가질 수 있으므로 동일한 브레이크 포인트에 대해 2 개의 엔트리가 동일한 시퀀스 넘버를 갖는 경우, 이들 엔트리에서 식별된 결정 엔진은 병렬로 인보크될 수 있다.

대 표 도


폴리시 엔진 1
20
폴리시 이네이블
애플리케이션
16
브레이크
포인트
구성 테이블
액션 유형
18
브레이크
포인트
14
10
애플리케이션 서버
12
20
폴리시 엔진 n

특허청구의 범위

청구항 1

삭제

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

청구항 6

삭제

청구항 7

삭제

청구항 8

삭제

청구항 9

삭제

청구항 10

삭제

청구항 11

삭제

청구항 12

삭제

청구항 13

애플리케이션의 동작 (behavior) 의 외부 실행-시간에 대한 적응을 허용하기 위한 시스템으로서,

결정 엔진 컴퓨팅 디바이스 상에서 동작하도록 적응된 적어도 하나의 결정 엔진으로서, 상기 적어도 하나의 결정 엔진은 애플리케이션 서버 상에 상주하는 애플리케이션과 상호작용하도록 구성되고, 상기 애플리케이션 서버는 상기 결정 엔진 컴퓨팅 디바이스로부터 분리된 서버 컴퓨팅 디바이스를 포함하고, 상기 적어도 하나의 결정 엔진은 상기 애플리케이션 외부에 있고, 상기 애플리케이션은 적어도 하나의 브레이크 포인트를 가지며, 상기 애플리케이션이 실행중인 동안에 상기 애플리케이션을 편집함이 없이 상기 적어도 하나의 결정 엔진이 변형되도록 구성되는, 상기 적어도 하나의 결정 엔진; 및

상기 애플리케이션에 동작 가능하도록 연결된 적어도 하나의 스토리지 엘리먼트로서, 상기 적어도 하나의 스토

리지 엘리먼트는 상기 적어도 하나의 브레이크 포인트에 대해 상기 적어도 하나의 결정 엔진의 적어도 하나의 식별자를 포함하는 브레이크 포인트 구성들의 리스트를 포함하도록 구성되고, 상기 애플리케이션은 상기 적어도 하나의 브레이크 포인트에서 상기 브레이크 포인트 구성들의 리스트에 액세스하도록 구성되고, 상기 브레이크 포인트 구성들의 리스트는 상기 애플리케이션이 실행중인 동안에 상기 적어도 하나의 결정 엔진의 상기 적어도 하나의 식별자의 변형을 허용하도록 구성되는, 상기 적어도 하나의 스토리지 엘리먼트를 포함하고,

상기 적어도 하나의 결정 엔진은, 상기 적어도 하나의 식별자를 이용하여 상기 애플리케이션이 상기 적어도 하나의 브레이크 포인트에 대해 상기 적어도 하나의 결정 엔진을 인보크 (invoke) 할 수 있도록 구성되며, 적어도 하나의 상기 브레이크 포인트 구성들의 리스트 및 상기 적어도 하나의 결정 엔진은, 상기 애플리케이션이 실행중인 동안에 상기 브레이크 포인트 구성들의 리스트 내의 상기 적어도 하나의 결정 엔진의 상기 적어도 하나의 식별자의 상기 변형 및 상기 적어도 하나의 결정 엔진의 상기 변형에 적어도 부분적으로 기초하여 상기 애플리케이션의 동작을 적응시키도록 구성되는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 시스템.

청구항 14

제 13 항에 있어서,

상기 스토리지 엘리먼트는 상기 애플리케이션 외부에 상기 브레이크 포인트 구성들의 리스트를 유지하도록 구성되는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 시스템.

청구항 15

제 13 항에 있어서,

상기 브레이크 포인트 구성들의 리스트는 상기 적어도 하나의 식별자에 대한 속성 맵핑 및 상기 적어도 하나의 식별자에 대한 액션 유형을 더 포함하는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 시스템.

청구항 16

제 13 항에 있어서,

상기 브레이크 포인트 구성들의 리스트는 상기 적어도 하나의 결정 엔진과 연관된 시퀀스 넘버를 상기 리스트 내에 더 포함하는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 시스템.

청구항 17

제 13 항에 있어서,

상기 브레이크 포인트 구성들의 리스트는 상기 적어도 하나의 브레이크 포인트를 대응하는 상기 적어도 하나의 결정 엔진으로 동적으로 맵핑하도록 구성되는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 시스템.

청구항 18

제 13 항에 있어서,

상기 애플리케이션 서버 상에 상주하는 상기 애플리케이션을 더 포함하는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 시스템.

청구항 19

제 18 항에 있어서,

상기 브레이크 포인트 구성들의 리스트는 상기 적어도 하나의 결정 엔진과 연관된 시퀀스 넘버를 상기 리스트 내에 더 포함하고,

상기 애플리케이션은 브레이크 포인트에 대해 존재하는 복수의 결정 엔진에 응답하여 병렬로 상기 복수의 결정 엔진을 인보크하도록 구성되고, 상기 복수의 결정 엔진에 대한 시퀀스 넘버는 동일한, 애플리케이션의 동작의

외부 실행-시간에 대한 적응을 허용하기 위한 시스템.

청구항 20

통신 시스템에서 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 방법으로서,

애플리케이션의 개별 브레이크 포인트들에서 상기 애플리케이션 외부의 브레이크 포인트 구성들의 리스트를 액세스하는 단계로서, 상기 브레이크 포인트 구성들의 리스트는 상기 개별 브레이크 포인트들에 대해 상기 애플리케이션 외부에 있는 적어도 하나의 결정 엔진의 적어도 하나의 식별자를 갖고, 상기 브레이크 포인트 구성들의 리스트는 상기 애플리케이션이 실행중인 동안에 상기 적어도 하나의 결정 엔진의 상기 적어도 하나의 식별자의 변형을 허용하도록 구성되고, 상기 애플리케이션이 실행중인 동안에 상기 애플리케이션을 편집함이 없이 상기 적어도 하나의 결정 엔진이 변형되도록 구성되는, 상기 리스트를 액세스하는 단계;

상기 적어도 하나의 식별자를 이용하여 상기 개별 브레이크 포인트들에 대응하는 상기 적어도 하나의 결정 엔진을 인보크 (invoke) 하는 단계; 및

상기 브레이크 포인트 구성들의 리스트의 상기 적어도 하나의 결정 엔진의 상기 적어도 하나의 식별자의 적어도 하나의 상기 변형 및 상기 적어도 하나의 결정 엔진에 적어도 부분적으로 기초하여 상기 애플리케이션의 동작을 적응시키는 단계를 포함하는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 방법.

청구항 21

삭제

청구항 22

제 20 항에 있어서,

상기 리스트를 액세스하는 단계는 상기 적어도 하나의 식별자에 대한 속성 맵핑 및 상기 적어도 하나의 식별자에 대한 액션 유형을 액세스하는 단계를 포함하는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 방법.

청구항 23

제 20 항에 있어서,

상기 리스트를 액세스하는 단계는 상기 리스트 내의 상기 적어도 하나의 브레이크 포인트에 대응하는 상기 적어도 하나의 결정 엔진과 연관된 시퀀스 넘버를 액세스하는 단계를 포함하는, 애플리케이션의 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 방법.

청구항 24

제 23 항에 있어서,

브레이크 포인트에 대해 존재하는 복수의 결정 엔진에 응답하여 병렬로 상기 복수의 결정 엔진을 인보크하는 단계를 더 포함하고, 상기 복수의 결정 엔진에 대한 상기 시퀀스 넘버들은 동일한, 애플리케이션 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 방법.

청구항 25

제 20 항에 있어서,

개별 브레이크 포인트들을 적어도 하나의 대응하는 결정 엔진으로 동적으로 맵핑하는 단계를 더 포함하는, 애플리케이션 동작의 외부 실행-시간에 대한 적응을 허용하기 위한 방법.

청구항 26

컴퓨팅 디바이스에 의한 실행에 응답하여, 상기 컴퓨팅 디바이스로 하여금 제 20 항 및 제 22 항 내지 제 25 항 중 어느 한 항에 기재된 방법을 수행하게 하는 명령들이 저장된 컴퓨터 판독가능 매체.

명 세 서

기 술 분 야

[0001] 본 발명은 일반적으로 통신, 정보 및 엔터테인먼트 서비스의 프로그래밍에 관한 것이다.

배 경 기 술

[0002] 전통 방식으로 기록된 소프트웨어 애플리케이션은 애플리케이션 코드 내에 정의된 기능성을 모두 갖는다. 이러한 애플리케이션이 전개되면, 애플리케이션을 제공하는 제공자들이 애플리케이션의 거동에 영향을 줄 여지가 거의 없다. 애플리케이션은 조정될 수 있는 약간의 파라미터를 가질 수도 있으나, 일반적으로 애플리케이션 코드가 변형되지 않는다면 애플리케이션의 거동은 고정된다.

[0003] 동적인 시장을 어드레싱하기를 원하는 서비스 제공자들은 새로운 서비스를 빠르게 제공하기 위한 유연성을 필요로 한다. 애플리케이션 코드를 변형하는 것과 연관된 사이클 시간은 프로세스를 허용할 수 없을 정도로 느리게 할 수 있다. 따라서, 소프트웨어 릴리즈 (release) 를 대기하지 않고 거동이 변형될 수 있는 애플리케이션 코드가 필요하다.

[0004] 유사한 동기가 전화망 서비스를 구축하기 위한 지능망 (Intelligent Network) 및 차세대 지능망 (Advanced Intelligent Network) 개념의 개발에 포함되었다. 전화기 스위치 내의 코드를 변경함으로써 새로운 서비스를 구축하는 것이 가능하지만, 이는 다소 느리고 귀찮다. 선택된 접근법은 2 개의 부분들 간의 접속을 확립하는데 있어서 전화기 스위치 소프트웨어의 상태를 특정한 콜 모델 (call model) 을 정밀하게 정의하는 것이다.
콜 모델의 다양한 포인트에서, 전화기 스위치는 외부 서비스 제어 포인트에게 어떻게 진행할 것인지에 대한 명령에 대해 질의하도록 구성될 수 있다. 새로운 서비스 기능성은 스위칭 시스템 자체 내의 소프트웨어에 대해 변화시키지 않고 서비스 제어 포인트에서의 로직의 거동을 변화시킴으로써 생성될 수 있다.

[0005] 이 접근법은 스위칭 시스템의 상태 및 기대된 거동을 특정하는 공통 콜 모델에 매우 의존한다. 이러한 콜 모델은, 콜을 확립하는 프로세스가 하나의 시스템으로부터 다른 시스템까지 일관되기 때문에 정의될 수 있다.
콜 모델의 정의 및 협의는 상당한 시간이 소요되고, 이는 전화기 콜의 프로세싱이 시간에 걸쳐 상대적으로 정적이기 때문에 허용 가능한 것이었다. 제공자들이 빈번하게 새롭고 변화된 서비스를 제공하는 환경에서, 서비스 로직의 내부 상태에 대해 이러한 상세한 모델을 정의하는 것이 가능하지 않을 것이다. 이 공통 콜 모델 기술이 바람직한 특성을 갖지만, 출현하는, 새롭고 변화된 서비스의 프로세싱에 직접 적용될 수 없다.

[0006] 통신 장비 판매자 및 운용자는 콜 프로세싱 시스템에 특징들을 추가하는데 있어서 유사한 문제를 어드레싱하였다. 가장 유력한 해결책, 차세대 지능망 (AIN) 에 대한 기초는 전화기 콜의 확립을 프로세싱하기 위한 기초로서 다른 공통 콜 모델을 정의한다. 이 공통 콜 모델은 외부 시스템이 결정에 대해 질의를 받을 수 있는 소정 세트의 이벤트를 제공한다. 예를 들어, Bunch 등의 미국 특허 5,940,487, "Programmable call processing system and method" 에서는 AIN 과 커플링된 분산형 통신 스위칭 시스템에 대한 이 접근법을 나타낸다. 콜을 프로세싱하는 서비스 스위칭 포인트들 (스위치들), 및 서비스 로직을 프로세싱하는 서비스 제어 포인트들의 분리는 스위치 내의 소프트웨어를 변화시키지 않고 새로운 콜링 서비스가 정의되는 것을 허용한다.
회선 교환망에서 전화기 콜링을 제어하기 위해 잘 작업된 이 접근법은 Parlay Group 사양 및 Java Community Process JAIN 사양과 같은 산업 포럼의 결과를 통해 다음 세대의 패킷 교환망으로 확장되었다. 이러한 표준화에 대한 필요성이 존재한다는 사실은 특정 기능성 (콜 제어) 을 다루고 기능에 대한 사전 동의 및/또는 표준화를 필요로 하는 이 접근법의 한계를 나타낸다. 따라서, 이 접근법은 시간에 걸쳐 정적인 애플리케이션 기능성에 적합하다.

[0007] 미국 특허 6,970,901, "Device and method for swapping out a part of a service logic program" 에서, Moritz 는 다수의 엔티티에 걸쳐 서비스 로직을 분산시키기 위한 메커니즘을 개시한다. Moritz 는 구체적으로, 클라이언트 단말기로 서비스 로직의 일부를 분산시킴으로써 클라이언트 단말기로서 작용하는 정보처리 기능을 가진 단말기 디바이스의 증가된 성능을 이용하는 것에 집중한다. Moritz 는 차징 (charging) 관련 정보를 결정하기 위해 이 분산된 접근법을 이용하는 방법을 기술한다. 클라이언트 디바이스로의 분산은 개인화 (personalization) 를 가능하게 할 수 있지만, 변화가 클라이언트 단말기 디바이스들 모두로 전파되게 해야 하기 때문에 전체 서비스의 거동의 변형을 용이하게 할 수 없게 한다. 통신 메커니즘 (Moritz 에서 "차지 티켓") 이 미리 정의되고 코딩될 수 있기 때문에 차징과 같은 단일의 토픽에 대한 분산을 가능하게 하는 것이 단순해진다. 따라서, Moritz 는 단지 매우 제한된 도메인에서 유연한 서비스 로직 프로그램에 대한 필요성을 어드레싱한다.

[0008] 미국 특허 6,967,957, "Architecture for the rapid creation of telephony services in a next generation network" 에서, Anjum 등은 "애플리케이션으로부터 하부의 콜-상태 관리 프로토콜 및 하드웨어의 상세를 숨기는" 객체 지향형 (object-oriented) 콜 모델을 설명한다. 구체적으로, Anjum 등은 회선 교환형 (circuit- switched) 및 패킷 교환형 (packet switched) 전화망 양자 모두에서 콜 제어를 나타내기에 충분히 추상적이도록 설계된 새로운 콜 모델을 설명한다. 이러한 콜 모델은, 그것이 기초가 되었던 JTAPI (Java Telephony Application Programming interface) 모델 보다 더 유연하지만, 여전히 단일의 기능성을 나타낸다.
따라서, 이 모델은, 거동이 또한 정의되지 않거나 미리 명확하게 알려지지 않은 더 일반적인 경우에 적용 가능하지 않다.

[0009] 드래프트 요건 문헌 "Policy Evaluation, Enforcement and Management Architecture"(OMA-AD-Policy_Evaluation_Enforcement_Management-V1_0-200600625-D) 에서 구체화된 바와 같이, OMA (Open Mobile Alliance) 내에서의 PEEM (Policy Evaluation Enforcement Management) 노력은, OMA 의 서비스 이네이블러 (enabler) 에 대한 지원으로서의 폴리시 (policy) 평가 및 실행에 대한 아키텍처를 설명한다. 이들 이네이블러는 그룹 리스트 관리, 메시징 및 로케이션과 같은 기능들을 포함한다. 아키텍처는 이들 이네이블러가 폴리시 결정에 대해 질의하기 위한 공통의 프레임워크를 제공하도록 설계된다. 시스템들 간의 맵핑 인터페이스의 프로세스를 단순하게 하기 때문에, 본 명세서에 설명된 폴리시 이네이블 프로그래밍을 위한 빌딩 블록으로서, 이러한 성능이 필요하지는 않다 하더라도 유용할 것이다. 그러나, OMA 작업은, 언제 폴리시 엔진 (PEEM 이네이블러) 에 질의할지, 또는 어느 폴리시 엔진에 질의할지를 이네이블러가 어떻게 결정하는지를 특정하지 않는다.

[0010] 다른 접근법은 워크플로우 (workflow) 시스템에 의해 지원되는 것이다. 복잡한 순서 프로세싱에 종종 적용되는 이들 시스템에서, 프로세싱 단계들 세트는 소프트웨어 코드의 방식으로 컴파일링되기 보다는 런 타임 (run time) 으로 해석되는 텍스트 방식으로 정의된다. 이는, 워크플로우는 워크플로우 엔진의 코드를 변화시키지 않고 변형될 수 있다는 점에서 유연성을 제공한다. 그러나, 해석 동작이 불충분하면, 순서 프로세싱에 적합하지만 실제 서비스의 실행에 일반적으로 불충분한 성능을 초래한다.

[0011] 데이터베이스 시스템은, 트리거 및 저장된 절차들이 로직을 실행하기 위해 호출되는 또다른 메커니즘을 제공하고, 이는 상당한 유연성을 제공한다. 그러나, 저장된 절차가 실행될 수 있는 동작들은 데이터베이스의 데이터에 대해 삽입, 삭제, 업데이트 및 선택하는 데이터베이스 동작에 대해서만 존재한다. 이 접근법의 문제는 데이터베이스 동작들에 대한 제약이다; 데이터 및 연관된 트리거 및 저장된 절차를 통해 네비게이팅하지 않는다면 전체 사양은 데이터베이스 시스템 내에 있고 쉽게 참고 가능하지 않거나 변형 가능하지 않다.

[0012] 따라서, 현재의 프로세스는 특정 유형의 소프트웨어 애플리케이션, 예를 들어 콜 프로세싱에 제한되고, 그들이 사용될 수 있기 전에 애플리케이션 프로세싱의 사전 동의 및/또는 표준화를 필요로 한다. 다른 현재의 프로세스는 데이터베이스 또는 워크플로우 시스템과 같은 미들웨어 (middleware) 내에만 존재하며, 이는 그들의 범위를 제한하고 서비스 실행을 위한 성능 요건을 충족시키지 않는다.

발명의 내용

해결하려는 과제

[0013] 따라서, 이러한 폴리시 엔진과 같은 외부 결정 포인트를 언제 그리고 어디서 질의할지를 동적으로 결정할 수 있는 시스템이 필요하다. 또한, 제어된 옵션 세트들로 구성될 수 있는 브레이크 포인트에 의해 특성화된, 이러한 동적 질의를 구성하기 위한 구조화되고 효율적인 메커니즘이 필요하다.

과제의 해결 수단

[0014] 본 발명은 애플리케이션 코드를 변화시키기 않고 실행 동안 애플리케이션의 거동이 적응 또는 구성되도록 허용하는 애플리케이션 소프트웨어를 구축 및 전개하기 위한 설계 아키텍처 및 방법 (methodology) 을 이롭게 제공한다. 애플리케이션의 구성 또는 적응에 의한 상이한 목적들을 위해 단일의 소프트웨어 애플리케이션의 다수 회 재사용이 제공된다. 또한, 본 발명의 시스템 및 방법은 그들이 소프트웨어 개발 사이클과 연관된 지연들 없이 전달될 수 있기 때문에 새로운 서비스 거동의 신속한 전개를 가능하게 한다.

[0015] 통신 시스템에서 애플리케이션의 애플리케이션 거동의 외부 실행-시간 적응을 허용하기 위한 시스템 및 방법은, 적어도 하나의 브레이크 포인트를 갖고 애플리케이션 서버 상에 상주하는 애플리케이션, 적어도 하나의 식별 가

능한 결정 엔진, 및 각 브레이크 포인트에 대해 결정 엔진의 적어도 하나의 식별자를 갖는 브레이크 포인트들의 리스트를 포함하여, 브레이크 포인트들 중 하나에서 애플리케이션은 브레이크 포인트들의 리스트를 액세스하고, 브레이크 포인트에 대응하는 결정 엔진의 리스팅된 인스턴스 (instance) 들을 인보크 (invoke) 하며, 결정 엔진에 기초하여 애플리케이션 거동을 적응시킨다.

[0016] 브레이크 포인트들의 리스트는 애플리케이션에 대해 외부 파일일 수도 있고, 각각의 식별자에 대한 속성 맵핑 및 각각의 식별자에 대한 액션 유형을 포함할 수 있으며, 또한 브레이크 포인트를 결정 엔진으로 동적으로 맵핑할 수 있다.

도면의 간단한 설명

[0017] 본 발명은 상세한 설명에 이어서 본 발명의 실시형태의 비 제한적인 예시의 방식에 의해 첨부된 도면을 참조하여 더 상세히 설명되며, 도면 전체에서 동일한 참조 부호는 동일한 부분들을 나타낸다. 그러나, 이해되어야 하는 바와 같이 본 발명은 도시된 정확한 배열 및 수단에 제한되지 않는다.

도 1 은 폴리시 이네이블 프로그램 실행을 위한 아키텍처이다.

도 2 는 브레이크 포인트 구성의 예시적 테이블이다.

도 3 은 브레이크 포인트 구성 테이블 내의 프로세싱 엔트리에서의 애플리케이션 거동을 나타내는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0018] 애플리케이션 거동의 외부 실행-시간 적응을 허용하기 위한 시스템 및 방법이 제시된다. 이 접근법의 주요 특징은, 컴파일링된 코드와 연관된 성능 및 결정 포인트와 애플리케이션 플로우 내의 가능한 액션을 정의하기 위한 유연성을 애플리케이션 개발자들에게 제공하고, 이들 결정 포인트의 외부 평가 엔진으로의 동적 맵핑을 허용하며, 단지 애플리케이션의 설계 및 개발 동안 보다는 애플리케이션이 개발됨에 따라 또는 애플리케이션이 실행중인 동안에도 애플리케이션 거동이 구성되는 것을 가능하게 하는 것이다.

[0019] 이러한 시스템 및 방법을 지원하는 2 개의 특정 메커니즘이 존재한다. 첫 번째는 브레이크 포인트 구성 테이블과 같은 외부 테이블이다. 어느 액션이 취해지는지를 결정하기 위해 실행 동안 그것이 정의된 브레이크 포인트에 도달할 때, 애플리케이션은 이 테이블을 판독한다. 이 테이블의 엔트리는, 소정의 브레이크 포인트에서 애플리케이션이 정의하고 있는 속성 또는 변수들에 대한 지식에 기초하여, 애플리케이션으로부터 별개로 정의될 수 있고, 애플리케이션이 실행되는 동안 변화될 수도 있다. 두 번째 적응 방법은 외부 시스템에 포함된다. 이들 시스템 내의 로직, 폴리시 및/또는 규칙은 후속하는 플로우 및 애플리케이션의 거동에 영향을 주는 응답을 생성한다. 이들 2 개의 메커니즘을 이용하여, 애플리케이션의 거동은 애플리케이션 코드를 재-기록 또는 변형하지 않고 의미있게 변형될 수 있다.

[0020] 도 1 은 본 발명의 시스템의 일 실시형태를 나타낸다. 폴리시 이네이블 애플리케이션 (policy enabled application; 10) 은 통신 시스템 (미도시) 의 애플리케이션 서버 (12) 내에 상주한다. 폴리시 이네이블 애플리케이션 (10) 은 애플리케이션 로직의 흐름 내에서 특정 세트의 구성 가능한 브레이크 포인트들 (14) 과 함께 구성된다. 애플리케이션 코드의 실행이 이들 브레이크 포인트들 (14) 중 하나에 도달할 때, 애플리케이션 코드는 액션 유형 (18) 을 포함하는 데이터를 포함한 브레이크 포인트 구성 테이블과 같은 외부 테이블 (external table; 16) 또는 브레이크 포인트 구성의 리스트를 참고한다. 이 데이터를 평가함으로써, 애플리케이션 (10) 은 브레이크 포인트 (14) 에서 취해질 액션을 결정할 수 있다. 테이블의 복잡성을 관리하기 위해서, 가능한 액션 유형의 수가 제한될 수 있다. 일반적인 액션은 애플리케이션이 외부 시스템 (20) 으로 데이터를 송신하고, 결과적인 응답을 이용하여 애플리케이션의 후속하는 액션을 결정하는 것이다. 이들 후속하는 액션은 연관된 성능을 이용하여 그 컴파일링된 코드 내에 정의된 바와 같이 애플리케이션 (10) 의 일 부분으로서 실행된다. 외부 시스템 (20) 은 애플리케이션 (10) 으로부터의 입력을 수용하고 정의된 하나 이상의 폴리시 세트를 적용하고, 응답을 제공하는 결정 엔진, 폴리시 엔진 또는 폴리시 결정 포인트일 수 있다. 애플리케이션의 거동은 애플리케이션의 코드로 변화시키는 것 대신에 하나 이상의 폴리시를 변형함으로써 변경된다. 다른 실시형태에서, 점프 테이블 (미도시) 이 애플리케이션 (10) 내에 구현될 수 있다. 이러한 경우, 애플리케이션 거동은 외부 결정 포인트의 프로세싱을 변형함으로써, 또는 애플리케이션 (10) 을 변형 및 다시 컴파일링함으로써 변화될 수 있다.

[0021] 외부 시스템 (20) 은 또한 워크플로우 시스템, 데이터베이스 시스템 또는 다른 소프트웨어 애플리케이션일 수도

있다. 맵핑이 상이한 소프트웨어 애플리케이션을 가르키도록 변화될 수 있다는 사실은 결정 포인트가 애플리케이션 (10) 에서 하드 코딩되는 (hard coded) 경우에도 유연성을 제공한다.

[0022] 폴리시 이네이블 애플리케이션 (10) 이 외부 폴리시 엔진 (20) 으로부터 수신하는 응답에 반응할 수도 있는 많은 방법이 존재한다. 그 후속하는 거동을 결정하는데 있어서 완전한 유연성을 외부 시스템 (20) 에 제공하는 애플리케이션 (10) 을 구축하는 것은 극히 어렵고 에러가 나기 쉬울 것이다. 그러나, 액션 유형 또는 옵션 (18) 의 범위에 대한 적절한 제한을 통해, 복잡성은 관리 가능한 수준에서 유지될 수 있다. 구체적으로, 애플리케이션 (10) 은 외부 시스템 (20) 에 의해 제공된 응답에 기초하여 제한된 수의 액션 유형 (18) 을 지원하도록 설계될 수 있다. 이들 액션 유형 (18) 의 예들은 포크 (fork) 또는 투-웨이 (two-way) 결정 포인트 액션, 널 (null) 액션, 및 변수 교체 액션을 포함할 수 있다.

[0023] 투-웨이 결정 액션 (18) 의 경우, 애플리케이션 (10) 은 외부 시스템 (20) 으로 메시지를 전송하고, 외부 시스템은 바이너리 (binary), 즉 "예/아니오" 또는 "진실/거짓" 응답을 제공한다. 애플리케이션 (10) 은 그 후, 응답이 "예" 인 경우 하나의 미리-정의된 경로를 그리고 응답이 "아니오" 인 경우 다른 경로를 따른다. 애플리케이션 (10) 은 이 결정 액션 (18) 에서 2 개의 옵션들 중 하나만을 수행하면서, 결정하기 위해서 외부 시스템 (20) 에 의해 이용된 폴리시 또는 프로세스에 기초하여 전체 거동이 광범위하게 변경될 수 있다. 이는, 다수의 경로가 가능한 멀티-웨이 (multi-way) 결정 액션으로 확장될 수 있다.

[0024] 일 예로서, 사용자에게 디지털 콘텐츠 아이템을 전달 또는 송신하도록 설계된 애플리케이션 (10) 은 콘텐츠를 송신하기 직전에 브레이크 포인트 (14) 를 포함할 수 있다. 이 브레이크 포인트 (14) 에 응답하여, 애플리케이션 (10) 은 질의 (query) 내의 콘텐츠 아이템의 리스트 및 사용자 양자 모두를 식별하는 정보를 외부 폴리시 엔진 (20) 에 공급한다. 애플리케이션 (10) 은, 응답이 "예" 인 경우 콘텐츠의 송신을 진행하고, 응답이 "아니오" 인 경우 송신을 취소하도록 프로그래밍된다. 일 상황에서, 외부 시스템 (20) 은, 사용자가 아이템 값을 지불하기 위한 충분한 잔액을 갖는지 여부를 결정하는 실시간 차징 엔진일 수 있다. 사용자의 잔액이 충분하면, 외부 시스템 (20) 은 사용자 계좌에서 인출하고 "예" 로 리턴한다. 잔액이 불충분하면, 외부 시스템 (20) 은 "아니오" 로 리턴한다. 다른 상황에서, 외부 시스템 (20) 은 예를 들어, 공동 보안 평가로서, 또는 식당이 방문자를 끌어들이기 위한 수단으로서, 소정의 물리적 로케이션 내의 사용자에게 콘텐츠가 다운로드되는 것을 단지 허용하는 인증 시스템일 수 있다. 사용자가 콘텐츠 아이템에 대해 특정된 로케이션 경계 내에 있으면, 외부 시스템 (20) 은 "예" 로 리턴하고, 그렇지 않으면 시스템 (20) 은 "아니오" 로 리턴한다. 따라서, 별개의 외부 시스템들 (20) 을 사용하는 2 개의 매우 상이한 서비스는 동일한 콘텐츠 전달 폴리시 이네이블 애플리케이션 (10) 을 이용한다.

[0025] 널 액션 (18) 의 경우, 애플리케이션 (10) 은 브레이크 포인트 (14) 를 간단히 수행하고 실행을 계속한다. 널 액션은, 애플리케이션이 외부 시스템 (20) 에 몇몇 이벤트를 간단히 알릴 필요가 있는 경우에 적합하다. 전술된 콘텐츠 다운로드 서비스 예에서, 애플리케이션 (10) 은 콘텐츠를 송신하기 직전에 널 액션 (18) 을 포함하는 브레이크 포인트 (14) 를 실행하도록 명령 받을 수 있다. 브레이크 포인트 (14) 를 실행하는 것은 외부 시스템 (20) 으로 하여금 콘텐츠 아이템의 리스트를 포함하는 다운로드 이벤트를 기록하게 하므로, 우편료 지불 계좌 (post paid account) 를 갖는 사용자에게 다음의 청구 사이클 동안 콘텐츠에 대해 차징될 수 있다.

[0026] 변수 교체 액션 (18) 의 경우에서, 애플리케이션 (10) 은 외부 시스템 (20) 으로부터의 응답을 이용하여 애플리케이션 (10) 에 의해 프로세싱되는 변수의 값을 변경한다. 상기의 콘텐츠 다운로드 서비스 예에서, 애플리케이션 (10) 은 콘텐츠를 송신하기 직전에 변수 교체 액션 (18) 을 포함하는 브레이크 포인트 (14) 를 실행하도록 명령 받을 수 있다. 브레이크 포인트 (14) 를 수행하는 것은 외부 시스템 (20) 으로 하여금 콘텐츠 아이템의 리스트를 필터링하게 하고, 아마도 식별된 사용자의 나이 또는 선호도에 기초하여 그 레이팅 정보가 사용자에게 용인 가능하지 않다는 것을 나타내는 아이템들을 제거한다. 외부 시스템 (20) 은 그 후, 콘텐츠 아이템의 초기 리스트를 필터링된 리스트로 교체하는 애플리케이션 (10) 으로 필터링된 리스트를 리턴하고, 다운로드를 진행한다.

[0027] 다른 액션 유형 (18) 은 애플리케이션 흐름의 제어에서 더 큰 유연성을 제공하도록 정의될 수 있다. 애플리케이션 (10) 을 구성하는데 있어서 심한 복잡성을 방지하기 위해서 액션 유형 또는 옵션 (18) 의 수를 제한하는 것이 바람직하다.

[0028] 폴리시 이네이블 애플리케이션 (10) 은 애플리케이션 서버 (12) 또는 서비스 전달 플랫폼 내에서 실행될 것이고, 각 브레이크 포인트 (14) 와 연관된 애플리케이션 거동을 특정하는 브레이크 포인트 구성 테이블 (16) 에 대한 로컬 액세스를 갖는다. 도 2 는 제품/서비스를 전개할 책임이 있는 제공 시스템에 의해 차지될 수

있는 예시의 테이블 (16) 을 나타낸다. 이 테이블 (16) 은 전술된 액션 유형 (18) 뿐만 아니라 이하에서 더욱 상세히 설명되는 다음의 필드들을 포함한다: 폴리시 엔진 또는 외부 시스템 (20) 에 대한 어드레스 (22), 시퀀스 넘버 (24) 및 속성 맵핑 (26). 폴리시-이네이블 애플리케이션 (10) 은 하나 이상의 외부 폴리시 결정 포인트들과 상호작용 한다. 구성 테이블 (16) 은 상호작용이 수행될 수 있는 프로그램 흐름 내의 포인트를 상호작용할 특정 결정 포인트 및 (속성이 송신되는 것과 같은) 상호작용하는 방식과 링크한다.

[0029] 본 명세서에서 폴리시 결정 포인트가 논의되었으나, 본 명세서에 정의된 메커니즘에서 외부 시스템 (20) 을 폴리시 평가자에 제한하지 않는다. 정의된 인보케이션 (invocation) 인터페이스를 갖는 임의의 외부 시스템 (20) 이 폴리시 결정 포인트 대신에 이용될 수 있다.

[0030] 간략화된 샘플 구성 테이블 (16) 이 표 1 에 도시된다. 이 경우, 브레이크 포인트 1 (14) 과 연관된 2 개의 브레이크 포인트 엔트리가 존재한다. 제 1 엔트리에 있어서, 애플리케이션 (10) 은 테이블 (16) 내의 시스템 어드레스 (22) 를 통해 속성들 (A, B 및 C) 을 폴리시엔진1 로 전송함으로써 속성 맵핑 (26) 을 수행하고, 응답을 이용하여 속성 C 의 값을 교체해야 한다. 제 2 엔트리에 있어서, 애플리케이션 (10) 은 테이블 (16) 내의 그 주소 (22) 를 통해 속성들 (26; A, C 및 D) 을 폴리시엔진1 로 전송하고, "예/아니오" 응답에 기초하여 결정 액션 (18) 을 수행해야 한다. 시퀀스 컬럼의 시퀀스 넘버 (24) 는 단일의 브레이크 포인트 (14) 와 연관된 인보케이션이 실행되는 순서를 나타낸다. 시퀀스 넘버 (24) 를 공유하는 인보케이션은 병렬로 실행될 수 있다. 시퀀스 넘버 (24) 가 이용되지 않는 경우, 애플리케이션 (10) 은 파일 내에 형성된 순서로 질의를 실행할 수 있다.

표 1

[0031] 브레이크 포인트 구성 테이블

브레이크 포인트 ID	브레이크 포인트 유형	시퀀스	시스템 어드레스	속성 맵핑	응답 교체
1	변수 교체	1	폴리시엔진1	A, B, C	C
1	포크	2	폴리시엔진1	A, C, D	
2	널	1	폴리시엔진2	X, Y, Z	

[0032] 도 3 은 도 1 에 도시된 애플리케이션 (10) 의 거동을 나타내는 흐름도를 포함한다. 도면의 좌측을 참조하면, 스텝 S1 에서 시작 시, 애플리케이션 (10) 은 스테이지 1 을 통해 그 설계된 활동을 수행한다. 스텝 S2 에서, 애플리케이션은 브레이크 포인트 1 (14) 에 도달한다. 이 포인트에서, 애플리케이션 (10) 은 브레이크 포인트 구성 테이블 (16) 을 참고하고, 브레이크 포인트 1 (14) 와 연관된 임의의 엔트리가 존재하는지를 본다.

[0033] 도 3 의 우측은 이들 엔트리를 프로세싱하기 위해 브레이크 포인트 1 모듈 내에서 이용되는 흐름을 나타낸다. 스텝 S3 진입 시, 애플리케이션 (10) 은 테이블 (16) 에 정의된 속성 맵핑 (26) 을 이용하여, 스텝 S4 에서 외부 시스템 (20) 을 인보크한다. 외부 시스템 (20) 이 응답할 때, 애플리케이션 (10) 은 스텝 S5 에서 프로세싱되는 브레이크 포인트의 유형 또는 액션 유형 (18) 을 결정한다. 액션 브레이크 포인트 유형 (18) 이 변수 교체이면, 애플리케이션 (10) 은 스텝 S6 에서 구성 테이블 (16) 에 정의된 바와 같이 교체를 수행하고, 스텝 S7 로 진행하여 프로세싱될 브레이크 포인트 엔트리 (14) 가 더 많은 지를 테스트한다. 브레이크 포인트 유형 (18) 이 널이면, 애플리케이션 (10) 은 응답을 대기할 필요가 없이 바로 스텝 S7 로 진행하여 더 많은 브레이크 포인트 엔트리 (14) 가 프로세싱될 필요가 있는지 테스트를 진행한다. 브레이크 포인트 유형 (18) 이 포크이면, 애플리케이션 (10) 은 스텝 S8 에서 외부 시스템 (20) 으로부터의 응답을 테스트한다. 외부 시스템 (20) 응답이 "예" 이면, 애플리케이션 (10) 은 스텝 S7 에서 더 많은 브레이크 포인트 엔트리 (14) 에 대한 테스트를 진행한다. 외부 시스템 응답이 "아니오" 이면, 애플리케이션은 임의의 추가의 브레이크 포인트 엔트리 (14) 의 프로세싱을 브레이크아웃하고, 스텝 S9 에서 "아니오" 엑시트 (exit) 상태로 메인 애플리케이션 흐름으로 리턴한다. 프로세싱될 브레이크 포인트 엔트리 (14) 가 더 없는 경우, 애플리케이션 (10) 은 스텝 S10 에서 "예" 엑시트 상태로 메인 애플리케이션 흐름으로 리턴한다.

[0034] 외부 시스템 (20) 과의 상호작용에서의 개념 및 에러 상태를 다루는 것은 도시되지 않는다. 애플리케이션 (10) 은 각종의 예외적 핸들링 거동으로 프로그래밍되어, 리턴된 에러 또는 응답이 없는 타임아웃 (time-out) 을 다룰 수 있다. 다르게는, 테이블이 특정의 핸들링된 예외의 경우를 고려하기 위한 액션을 나타내도록 확

장될 수 있다. 애플리케이션 (10) 이 이들 상황을 다루는 방식은 이 메커니즘에 제한되지 않는다.

[0035] 동적 프로그래밍 접근법의 주요 양태는, 애플리케이션 브레이크 포인트 (14) 가 결정 포인트 시스템 (20) 및 그 안의 특정 폴리시로 동적으로 맵핑될 수 있다는 것이다. 이 맵핑이 수행될 수 있는 수많은 방법이 존재한다. 아마도 가장 좋은 직시 (straightforward) 는 수동의 접근법이며, 여기서 분석가는 브레이크 포인트 구성 테이블 (16) 을 생성하고, 각 브레이크 포인트 (14) 에 대응하는 폴리시를 정의하고, 구성 테이블 (16) 을 애플리케이션 서버 (12) 안에 로딩하며, 폴리시를 결정 포인트 안에 로딩한다. 애플리케이션 브레이크 포인트 (14) 및 폴리시가 구조화된 방식으로 가시화될 수 있는 경우 이 맵핑 프로세스의 더 큰 정도의 자동화 및 확인이 가능하다.

[0036] 설계 환경은 사용자에 의해 선택된 애플리케이션 (10) 에서 모든 브레이크 포인트들 (14) 의 구조화된 표현을 개입시킬 수 있다. 환경은 그것에 알려진 결정 포인트로부터 폴리시를 검색할 수 있다. 사용자가 표 2 에 도시되는 간단한 표현 및 브레이크 포인트 (14) 를 선택할 때, 환경은 유효한 이들 폴리시 만을 검색 및 제시할 수 있다. 예를 들어, 브레이크 포인트 (14) 가 포킹 (forking)_동작 또는 투-웨이 결정 포인트 (18) 만을 허용하는 경우, 예/아니오 또는 참/거짓 값들을 리턴하는 이들 폴리시만이 적합하다. 폴리시를 선택한 후에, 표 3 에 도시된 바와 같이, 사용자는 선택된 브레이크 포인트 (14) 에서 소프트웨어 애플리케이션 (20) 으로부터 이용 가능한 변수들을 폴리시에 대한 인터페이스로 특정된 것들로 맵핑한다. 환경은 그 후, 예를 들어 유형들 (예를 들어, 스트링 (string), 정수 등) 이 매치되는 것을 확보하는 맵핑을 유효하게할 수 있다.

이 방식으로 모든 맵핑들이 성공적으로 정의된 후에, 환경은 대응하는 브레이크 포인트 구성 테이블 (16) 을 생성하고, 그것을 애플리케이션 서버 (12) 상에 인스톨 할 수 있다.

표 2

[0037] 간략화된 브레이크 포인트 표현 구조

브레이크 포인트
명칭
설명
허용된 유형 (널, 변수 교체, 포크)
이용 가능한 변수들 (명칭 및 유형)
…

표 3

[0038] 간략화된 폴리시 인보케이션 인터페이스 구조

폴리시
명칭
설명
입력 [변수 1 (명칭/유형), 변수 2 …]
출력 [변수 1 (명칭/유형), 변수 2 …]
…

[0039] 본 발명은 특정 실시형태로 설명되었으나, 본 발명이 이러한 실시형태에 제한되는 것으로서 해석되어서는 안되고, 차라리 이하의 청구범위에 따라 해석되어야 한다.

부호의 설명

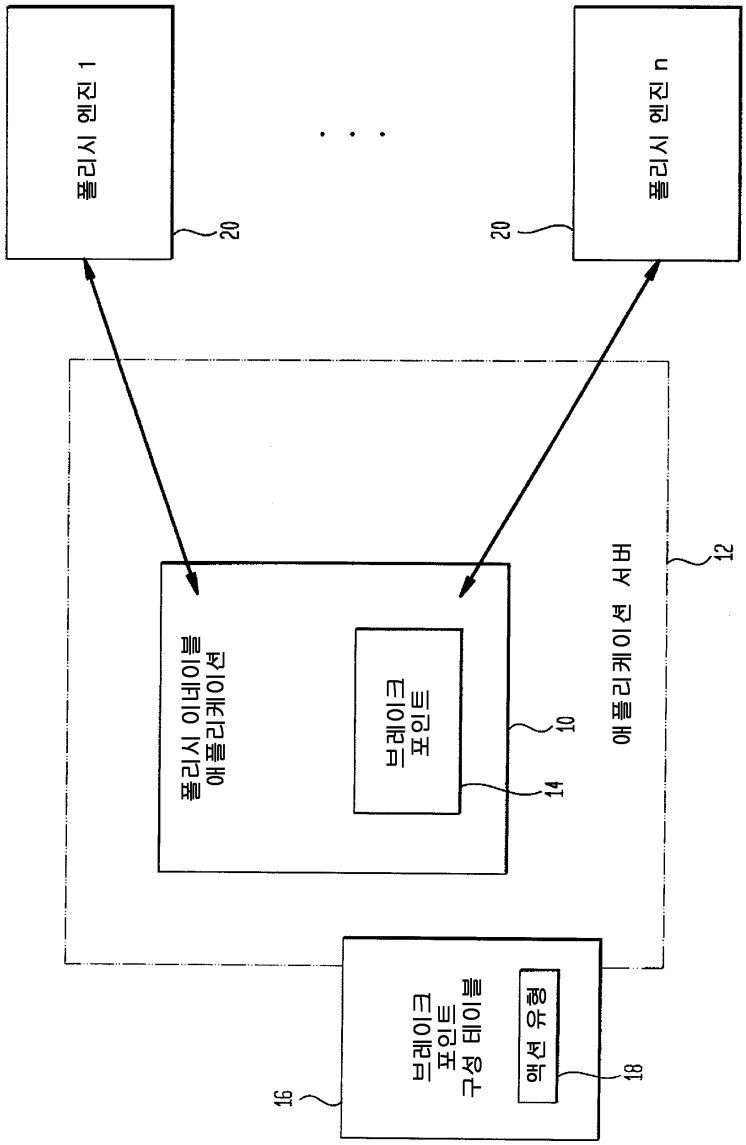
[0040] 10: 폴리시 이네이블 애플리케이션 12: 애플리케이션 서버

14: 브레이크 포인트 16: 브레이크 포인트 구성 테이블

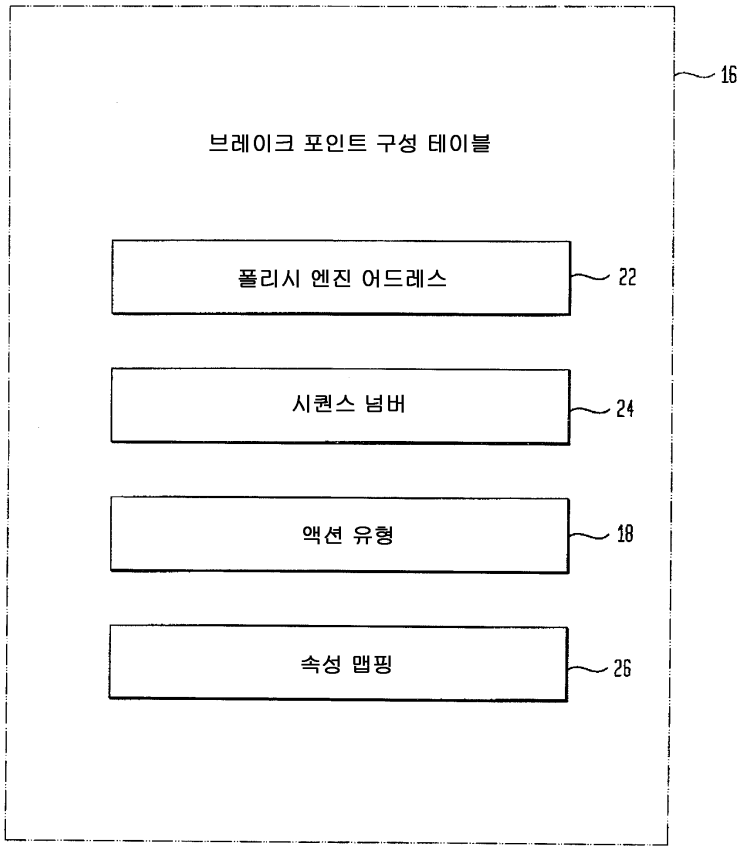
18: 액션 유형 20; 폴리시 엔진

도면

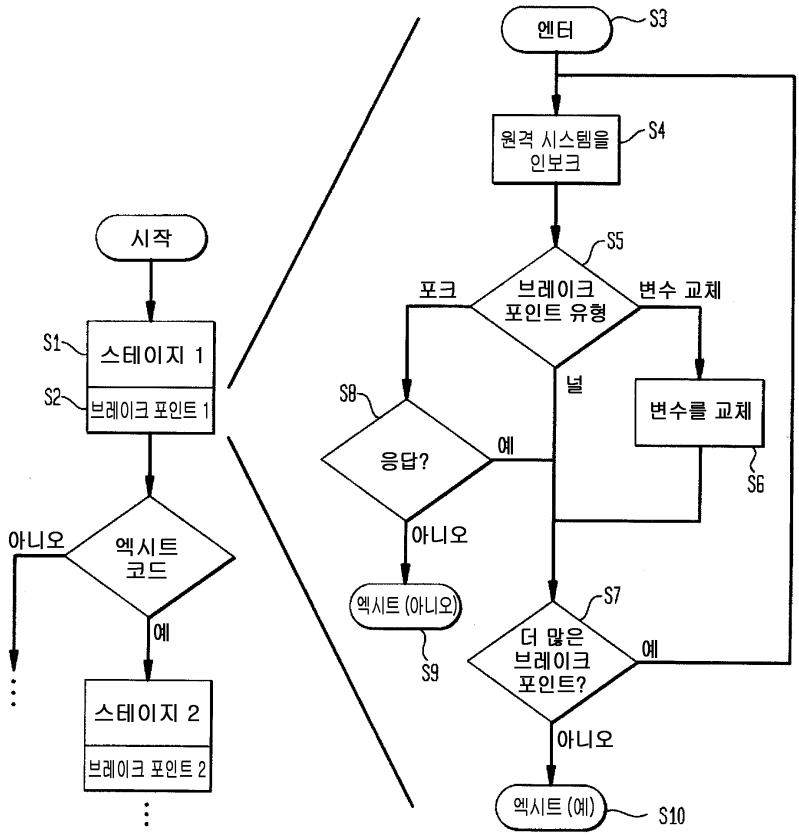
도면1


폴리시 엔진 1
20
. . .
폴리시 엔진 n
20
애플리케이션 서버
12
폴리시 이네이블
애플리케이션
브레이크
포인트
10
14
브레이크
포인트
구성 테이블
액션 유형
16
18

도면2



도면3


시작
S1
스테이지 1
S2
브레이크 포인트 1
아니오
엑시트 코드
예
스테이지 2
브레이크 포인트 2
엔터
S3
원격 시스템을 인보크
S4
S5
포크
브레이크 포인트 유형
변수 교체
널
S8
응답?
예
아니오
변수를 교체
S6
엑시트 (아니오)
S9
S7
더 많은 브레이크 포인트?
예
아니오
엑시트 (예)
S10