(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0098010
(43) 공개일자 2008년11월06일

(51) Int. Cl.
G06F 15/16 (2006.01) *G06F 15/163* (2006.01)
G06F 9/06 (2006.01)
(21) 출원번호 10-2008-7018415
(22) 출원일자 2008년07월25일
심사청구일자 없음
번역문제출일자 2008년07월25일
(86) 국제출원번호 PCT/US2007/002671
국제출원일자 2007년01월30일
(87) 국제공개번호 WO 2007/092231
국제공개일자 2007년08월16일
(30) 우선권주장
11/347,440 2006년02월03일 미국(US)

(71) 출원인
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
셰켈, 제리 제이.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
얀센, 라이언 엠.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
골드스테인, 세쓰 디.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
(74) 대리인
양영준, 백만기

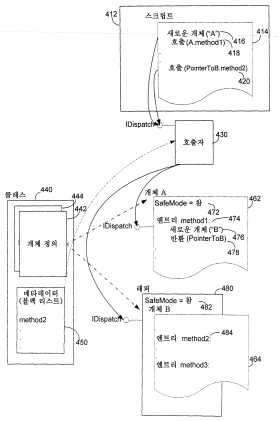
전체 청구항 수 : 총 20 항

(54) 컴퓨터 실행가능 컴포넌트들을 갖는 컴퓨터 판독가능 매체,및 메소드를 갖는 개체로 프로그램된 컴퓨터 시스템을동작시키는 프로세스

(57) 요 약

본 발명은 개체 상에 배치된 메소드 호출을 인터셉트하도록 구성된 컴퓨터 시스템에 관한 것이다. 메소드 호출을 인터셉트함으로써, 각 메소드 호출마다 처리가 실행될 수 있다. 처리의 일부로서, 개체를 위한 메타데이터는 그 개체를 위한 메소드들을 부분집합으로 나누기 위해 참고된다. 임의의 특정 메소드 호출에 응답하여 실행된 처리는 메소드가 속하는 부분집합에 기초하여 행해질 수 있다. 메타데이터의 유형은 컴퓨터 시스템의 원하는 동작에 따라 다를 수 있다. 화이트 리스트 또는 블랙 리스트를 나타내는 메타데이터는 안전하다고 간주된 메소드로의 액세스를 허용하거나 또는 안전하지 않다고 간주된 메소드로의 액세스를 거부하는 보안 구역을 구현하는 컴퓨터 시스템에서 사용될 수 있다. 성능 모니터링 시스템에서, 메타데이터는 로그될 메소드, 또는 실행 비용이 평가될 메소드를 식별할 수 있다. 메소드 호출에 의해 반환된 값은 또한, 메소드 호출이 인터셉트될 때 처리될 수 있다. 반환된 값은 보안 구역의 구현시에 유용하거나 또는 다른 처리를 실행하기에 유용한 데이터를 보유하는 래퍼로 래핑될 수 있다.

대 표 도 - 도4



특허청구의 범위

청구항 1

컴퓨터 실행가능 컴포넌트들을 갖는 컴퓨터 판독가능 매체에 있어서,

a) 다수의 개체 - 각 개체는 하나의 메소드를 포함함-; 및

b) 인터페이스 컴포넌트

를 포함하고,

상기 b) 인터페이스 컴포넌트는

i) 다수의 개체 중의 임의의 개체 내의 호출된 메소드에 대한 호출을 수신하고;

ii) 상기 호출의 수신에 응답하여 처리를 실행하며;

iii) 상기 호출된 메소드를 호출하도록

적응된 컴퓨터 판독가능 매체.

청구항 2

제1항에 있어서, 상기 다수의 개체의 각각은 ActiveX 개체를 포함하고, 상기 인터페이스 컴포넌트는 IDispatch 인터페이스를 구현하는 컴퓨터 판독가능 매체.

청구항 3

제1항에 있어서, 상기 인터페이스 컴포넌트는

iv) 상기 호출된 메소드가 실행되어야 하는지 판정을 하고;

v) 상기 판정에 응답하여 상기 호출된 메소드를 선택적으로 호출하도록

더욱 적응되는 컴퓨터 판독가능 매체.

청구항 4

제3항에 있어서, 상기 호출된 메소드를 선택적으로 호출하는 것은 상기 호출된 메소드가 호출되지 않아야 한다는 것을 나타내는 판정에 응답하여 예외를 생성하는 것을 포함하는 컴퓨터 판독가능 매체.

청구항 5

제1항에 있어서,

c) 호스트 컴포넌트; 및

d) 운영 체제 컴포넌트

를 더 포함하고,

상기 인터페이스 컴포넌트는 상기 호스트 컴포넌트의 일부이고, 상기 다수의 개체의 최소한 일부는 운영 체제 컴포넌트를 액세스하는 컴퓨터 판독가능 매체.

청구항 6

제5항에 있어서, 상기 호스트 컴포넌트는 웹 브라우저이고, 상기 운영 체제 컴포넌트는 파일 관리 시스템을 포함하는 컴퓨터 판독가능 매체.

청구항 7

제6항에 있어서, 상기 인터페이스 컴포넌트는

i) 상기 호출된 메소드가 신뢰할 수 없는 코드로부터 직접 또는 간접적으로 호출되었는지 판정하는 처리를 실행

하고;

ii) 상기 호출된 메소드가 신뢰할 수 없는 코드로부터 호출되었으면, 상기 호출된 메소드가 신뢰할 수 없는 코드에 의한 실행에 적합한 것으로 간주되었는지 판정하는 처리를 실행하며;

iii) 상기 처리의 결과들에 기초하여 상기 호출된 메소드에 대한 호출을 선택적으로 허용하도록

적응되는 컴퓨터 판독가능 매체.

청구항 8

메소드를 갖는 개체로 프로그램된 컴퓨터 시스템을 동작시키는 프로세스에 있어서,

a) 메소드에 관한 정보를 개체와 관련시키는 단계;

b) 상기 메소드에 대한 호출시에, 상기 정보를 참고하는 단계; 및

c) 상기 정보에 기초하여 상기 호출을 선택적으로 처리하는 단계

를 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 9

제8항에 있어서,

i) 상기 개체가 다수의 메소드를 갖고;

ii) 정보를 상기 개체와 관련시키는 단계는 제1 시나리오에서 실행가능한 메소드들의 부분집합을 식별하는 단계를 포함하며;

iii) 상기 호출을 선택적으로 처리하는 단계는 상기 메소드가 상기 부분집합 내에 있을 때 호출을 허용하는 단계를 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 10

제9항에 있어서, 상기 호출을 선택적으로 처리하는 단계는 상기 메소드가 상기 부분집합 내에 없을 때 예외를 생성하는 단계를 더 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 11

제8항에 있어서,

i) 상기 개체가 다수의 메소드를 갖고;

ii) 정보를 상기 개체와 관련시키는 단계는 제1 시나리오에서 실행 불가능한 메소드들의 부분집합을 식별하는 단계를 포함하며;

iii) 상기 호출을 선택적으로 처리하는 단계는 상기 메소드가 상기 부분집합 내에 없을 때 호출을 허용하는 단계를 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 12

제8항에 있어서, 상기 호출을 선택적으로 처리하는 단계는 상기 메소드에 대한 제2 정보를 선택적으로 저장하는 단계를 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 13

제12항에 있어서, 상기 메소드에 대한 제2 정보를 선택적으로 저장하는 단계는 상기 메소드의 예외 비용의 표시자(indicator)를 저장하는 단계를 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 14

제12항에 있어서, 상기 메소드에 대한 제2 정보를 선택적으로 저장하는 단계는 상기 메소드가 실행되었다는 표시자를 저장하는 단계를 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 15

제8항에 있어서,

i) 상기 메소드가 반환 값을 생성하고;

ii) 상기 호출을 선택적으로 처리하는 단계는 상기 반환 값을 위한 래퍼(wrapper)를 선택적으로 생성하는 단계를 포함하는 컴퓨터 시스템 동작 프로세스.

청구항 16

컴퓨터 판독가능 매체에 있어서,

a) 제1 데이터 구조; 및

b) 컴퓨터 실행가능 컴포넌트들

이 저장되어 있는데,

상기 a) 제1 데이터 구조는

i) 제1 개체의 정의를 포함하는 최소한 하나의 필드; 및

ii) 제2 개체의 정의를 포함하는 최소한 하나의 필드

를 포함하고,

상기 b) 컴퓨터 실행가능 컴포넌트들은

i) 상기 제1 개체 정의로부터 인스턴스화된 제1 개체 - 상기 제1 개체는 상기 제2 개체의 정의로부터 제2 개체를 인스턴스화시키도록 적응된 컴퓨터 실행가능 명령어들과 관련됨-;

ii) 상기 제2 개체 정의로부터 인스턴스화된 제2 개체; 및

iii) A. 상기 제2 개체에 대한 호출을 수신하고;

B. 상기 제2 개체에 대한 호출을 배치하며;

C. 상기 제2 개체가 인스턴스화되었다는 동작 상태의 표시를 저장하도록 적응된 래퍼

를 포함하는 컴퓨터 판독가능 매체.

청구항 17

제16항에 있어서, 상기 래퍼는 상기 제2 개체를 식별하는 호출을 수신하여 제2 개체 내의 메소드를 호출하도록 적응된 호출 컴포넌트를 포함하는 컴퓨터 판독가능 매체.

청구항 18

제17항에 있어서, 상기 제1 개체는 다수의 메소드를 포함하고,

상기 컴퓨터 판독가능 매체는

c) 보안 구역에서 동작가능한 상기 제1 개체 내의 메소드들을 식별하는 최소한 하나의 필드를 갖는 제2 데이터 구조

를 더 포함하는 컴퓨터 판독가능 매체.

청구항 19

제17항에 있어서, 상기 호출 컴포넌트는 상기 호출이 상기 제2 데이터 구조 내에서 식별된 메소드에 대한 것일 때 상기 래퍼 상에 호출을 선택적으로 배치하도록 적응되는 컴퓨터 판독가능 매체.

청구항 20

제16항에 있어서,

a) 상기 제2 개체는 보안 구역 내에서의 동작을 위한 동작 상태를 갖고;

b) 상기 제2 개체는 상기 보안 구역 동작 상태에서 상기 제2 개체를 액세스하는 인터페이스를 가지며,

c) 상기 래퍼는 상기 인터페이스를 통해 상기 제2 개체 상에 호출을 배치하도록 적응되는

컴퓨터 판독가능 매체.

명 세 서

배 경 기 술

<1> 다수의 소프트웨어 시스템에서, 소정의 개체로의 액세스는 이러한 개체들이 호출되는 상황에 기초하여 제한될 수 있다. Microsoft Corporation에 의해 제공된 INTERNET EXPLORER® 웹 브라우저와 같은 웹 브라우저는 "보안 구역"을 구현하기 위해 소정의 개체로의 액세스를 제한할 수 있다. 인터넷에서 다운로드한 스크립트는 보안 구역에서 실행되도록 허용될 수 있지만, 이들 스크립트는 웹 브라우저가 설치되는 컴퓨터의 동작을 중단시킬 수 있는 개체로의 액세스가 거부될 수 있다. 보안 구역을 가지면, 컴퓨터 사용자가 인터넷 또는 다른 신뢰할 수 없는 소스로부터 스크립트를 다운로드할 수 있게 하는 한편으로, 그러한 다운로드된 스크립트가 부주의로 인해 부적절하게 작성되었거나 또는 컴퓨터의 동작을 중단시키는 방식으로 악의적으로 작성되었기 때문에 컴퓨터상에서 실행될 때 손상을 줄 수 있는 위험성도 감소시킬 수 있다.

<2> ActiveX® 개체는 액세스가 제한될 수 있는 개체의 한 예이다. 스크립트를 실행하도록 구성된 컴퓨터는 다수의 ActiveX® 개체를 포함할 수 있다. 각각의 ActiveX® 개체는 하나 이상의 인터페이스를 포함한다. 각각의 인터페이스는 인터페이스를 통해 호출될 수 있는 메소드를 가질 수 있어서, 스크립트가 이들 메소드를 호출하여 스크립트 실행 동안 기능을 실행하게 할 수 있다. 그러한 기능은 컴퓨터 디스플레이 상에서의 정보의 표시, 이메일을 송신하기 위한 이메일 시스템의 액세스, 또는 컴퓨터의 파일 관리 시스템에서의 파일의 변경을 포함할 수 있다.

<3> 컴퓨터의 디스플레이 상에 출력을 표시하는 것과 같은, ActiveX®에 의해 실행된 몇몇 기능은 일반적으로 양호한 것일 수 있다. 그러한 양호한 기능을 실행하는 메소드만을 노출시키는 개체 상에서의 보안 구역으로부터의 호출은 허용될 수 있다. 그러나, 파일 관리 시스템을 액세스하는 것과 같은 기타 기능은 부적절하게 또는 악의적으로 스크립트 내에서 사용된 경우에 컴퓨터에 손상을 줄 수 있다. 보안 구역을 실현할 때, 그러한 유해한 기능을 실행하는 메소드를 노출시키는 개체 상에서의 호출은 허용되지 않는다.

<4> 보안 구역으로부터 호출될 때 소정의 개체로의 액세스를 제한하는 메커니즘이 제공된다. ActiveX® 개체의 경우에, 이 메커니즘은 IObjectSafety라고 불리는 선택 제한 인터페이스이다. 이 인터페이스를 구현하는 ActiveX® 개체는 동작의 "안전 모드"를 갖고 있어서 안전 모드에서 인스턴스화될 수 있다는 것을 나타내기 위해 스크립트를 실행하는 호스트 프로그램으로부터의 쿼리에 응답할 수 있다. 일단 안전 모드에서 인스턴스화되면, 호출은 그 인터페이스들 중의 임의의 것을 통해 개체 상에서 이루어질 수 있다. 안전 모드에서 실행될 때는, 안전하지 않은 메소드가 확실하게 호출되지 않게 하는 것이 ActiveX® 개체의 책임이다. 이 특징은 동작의 안전 모드를 지원하는 ActiveX® 개체의 인스턴스만을 생성함으로써, 그리고 그러한 모든 개체를 안전 모드에서 인스턴스화함으로써, 웹 브라우저가 보안 구역을 구현할 수 있게 한다.

발명의 상세한 설명

<5> 컴퓨터 시스템의 소프트웨어는 개체로의 액세스를 제어하는 개선된 프로세스를 제공함으로써, 더욱 강력하고, 더욱 용이하게 개발될 수 있으며, 제한하고자 하는 개체로의 액세스가 보다 덜 승인되게 할 가능성이 있다. 양호하게, 프로세스는 프로세스 실행을 위해 구체적으로 구현될 것을 개체에 요구하지 않는데, 이것은 프로세스의 사용을 단순화하고, 개선된 프로세스에 따라 현존하는 개체가 어느 컴퓨터 시스템에서나 사용될 수 있게 한다.

<6> 한 실시양상에서, 본 발명은 메소드 호출을 인터셉트할 수 있어서, 실행 전이나 후에 메소드 호출이 처리될 수 있게 하는 소프트웨어 시스템에 관한 것이다. 호출은 메소드 호출을 수신하고, 호출을 허용하거나 거부하기 전에 수신하는 각 호출을 처리하는 인터페이스 컴포넌트를 제공함으로써 인터셉트될 수 있다. 이 기술은 호출된 메소드가 실행하기에 안전한 것으로 간주되는지 판정하기 위해 보안 구역으로부터 배치된 각 호출을 처리함으로

써 보안 구역을 구현하는데 사용될 수 있다. 결과적으로, 개체로의 액세스는 각 메소드 호출마다 제한될 수 있다.

<7> 다른 실시양상에서, 본 발명은 메소드를 호출을 선택적으로 처리하기 위한 컴퓨터 시스템의 동작에 관한 것이다. 프로세스는 컴퓨터 시스템 내에 프로그램된 메소드에 대한 저장된 정보를 사용한다. 메소드가 호출될 때, 저장된 정보는 호출 처리시에 사용하기 위해 액세스된다. 프로세스는 어느 메소드 호출이 거부되어야 할지 허용되어야 할지를 저장된 정보가 나타내면 보안 구역의 구현의 일부로서 사용될 수 있다. 프로세스는 또한, 선택된 메소드의 실행을 로그하거나 또는 프로그램 실행 비용에 관한 통계를 누적하는 시스템의 일부로서와 같은 상황을 포함하는 기타 상황에서 적용될 수 있다.

<8> 다른 실시양상에서, 본 발명은 특별한 처리가 요구되는 구역으로부터 간접적으로 인스턴스화된 개체를 식별하기 위한 메커니즘을 포함하는 컴퓨터의 소프트웨어에 관한 것이다. 한 구역으로부터 간접적으로 인스턴스화된 개체를 추적하기 위해, 개체는 관련된 래퍼(wrapper)를 가질 수 있다. 래퍼는 개체가 구역으로부터 직접 또는 간접적으로 인스턴스화되었는지 나타내는 정보를 저장할 수 있다. 래퍼는 개체가 다른 개체를 인스턴스화하는 프로세스에서 사용될 수 있다. 한 개체가 또 다른 개체를 인스턴스화할 때, 각 개체는 프로세서로부터 값을 물려받는데, 이 값은 래퍼 내에 저장된다. 개체의 체인이 형성될 때, 각 개체는 체인 내의 제1 개체와 동일한 값을 갖는다. 개체가 구역 내에서 인스턴스화되었다는 것을 나타내는 값이 체인 내의 루트 개체에 주어지면, 체인 내의 모든 개체는 동일한 값을 가져서, 그 구역으로부터 인스턴스화된 이들 개체를 식별할 것이다. 개체로의 액세스를 제한하는 프로세스의 일부로서 이들 저장된 표시를 사용함으로써, 보안 구역을 구현하는 코드 개발자의 부담이 감소되는데, 그 이유는 코드 개발자가 간접적으로 인스턴스화될 수 없는 개체를 식별할 필요가 없거나, 또는 보안 구역에서 실행하기에 안전하다고 생각되지 않는 그들 개체로의 액세스를 제한할 필요가 없기 때문이다.

<9> 상기 설명은 본 발명의 비제한적인 요약이고, 본 발명은 첨부된 청구범위에 의해 정의된다.

실 시 예

<15> 우리는 개체로의 액세스를 제어하는 프로세스가 개선된 경우에 보안 구역의 구현이 개선될 수 있다는 것을 알아냈다. 개체가 안전한 인터페이스를 구현할 때 보안 구역 내에서부터 개체를 인스턴스화하는 것만이 공지되어 있지만, 보안 구역에서 실행되는 스크립트가 보안 구역 내에서부터 인스턴스화되지 않은 개체로의 액세스를 부여받은 경우에는 이러한 예방책도 적절하지 않을 수 있다. 이 상태는 개체가 또 다른 개체를 반환하는 경우에 발생할 수 있다. 예를 들어, 안전한 인터페이스를 구현하는 제1 개체는 그 메소드들 중의 하나가 호출될 때, 제2 개체를 인스턴스화하고, 제2 개체를 스크립트에 반환할 수 있다. 제2 개체가 "안전"한 것으로 간주되지 않으면, 스크립트는 간접적으로 안전하지 않은 개체로의 액세스를 얻는다.

<16> 이러한 결과를 방지하기 위해, 각 메소드의 안전은 각 메소드 호출마다 검증될 수 있다. 호출이 이루어지면, 호출은 호출되고 있는 메소드가 안전하다고 간주되는 경우에만 진행될 수 있게 된다. 아래에 더욱 상세하게 설명되는 바와 같이, 본 발명의 실시예에서, 그러한 보안 구역은 메소드 호출을 인터셉트함으로써 구현될 수 있다. 메소드 호출이 보안 구역 내에서부터 이루어지면, 호출된 메소드는 안전하다고 간주되는 경우에 진행될 수 있게 되지만, 메소드가 안전한 것으로 간주되지 않는 경우에 액세스가 차단된다.

<17> 부수적으로, 보안 구역에서 실행되는 스크립트로부터 간접적으로 인스턴스화된 개체를 식별하는 메커니즘이 제공될 수 있다. 몇몇 실시예에서, 이 메커니즘은 개체가 인스턴스화될 때 개체로의 특성의 할당을 수반할 수 있다. 각 개체가 인스턴스화될 때, 각 개체는 자신을 인스턴스화한 개체로부터 이 특성을 물려받을 수 있다. 보안 구역에서 직접적으로 인스턴스화된 개체에 인식가능한 이 특성 값을 제공하는 것은 보안 구역에서 간접적으로 인스턴스화된 임의의 개체가 이 값을 물려받게 한다. 그러한 메커니즘의 특정 예로서, 래퍼는 보안 구역에서 간접적으로 인스턴스화된 개체를 위해 제공된다. 래퍼는 개체의 정의에 대한 어떤 변경도 요구하지 않고, 개체의 특성을 정의하는 정보를 저장한다.

<18> 도 1을 참조하면, 컴퓨터 시스템(110)의 블록도가 도시된다. 도 1의 블록도는 매우 단순화된 것으로, 본 발명의 예시적인 실시예와 관련하여 설명된 개체로의 액세스 제어에 관련된 컴퓨터 시스템의 그러한 실시양상만을 도시한 것이다.

<19> 컴퓨터 시스템(110)은 인터넷(120)으로의 액세스를 갖는 것으로 도 1에 도시된다. 인터넷(120)을 통해, 컴퓨터 시스템(110)은 서버(130)로 액세스할 수 있다. 컴퓨터 시스템(110)은 서버(130)로부터 스크립트를 포함한 정보를 다운로드할 수 있다. 여기에서 사용된 바와 같이, "스크립트"라는 용어는 하나 이상의 컴퓨터 실행가능 명

령어의 컬렉션을 칭하는 것이다. 일반적으로, 스크립트는 단일 함수 또는 밀접하게 관련된 함수들의 그룹을 실행하기 위해 컴퓨터 시스템에 제공되는 비교적 작은 프로그램이다. 종종, 스크립트는 웹 페이지를 애니메이션으로 하거나 사용자를 위한 더욱 상호작용적인 경험을 만들어내기 위해 웹 페이지를 액세스하는 클라이언트 컴퓨터에 웹 페이지에 의해 제공된다. 그러나, 본 발명은 임의의 특정 크기, 함수 또는 용도의 스크립트에 제한되지 않는다. 본 발명은 또한 인터넷을 통해 다운로드된 스크립트에도 제한되지 않는다.

<20> 컴퓨터 시스템(110)은 스크립트(114)를 실행할 수 있는 호스트(112)를 포함한다. 여기에서, 호스트(112)는 스크립트를 실행하기 위해 종래의 기술에 따라 구성될 수 있는 소프트웨어 컴포넌트이다. 호스트(112)는 때때로 "인터프리터", "JIT(just in time) 컴파일러" 또는 "가상 기계"로 불리는 소프트웨어를 포함할 수 있다. 호스트(112)는 스크립트를 실행하는 것 이외의 기능을 실행할 수 있다. 인터넷으로부터 다운로드된 스크립트의 예에서, 호스트(112)는 웹 브라우저일 수 있다. 그러나, 임의의 적합한 호스트가 사용될 수 있다.

<21> 도시된 실시예에서, 스크립트(114)는 실행을 위해 호스트(112)에 의해 해석되는 컴퓨터 실행가능 명령어를 포함한다. 스크립트(114)의 실행 단계는 컴퓨터 시스템(110) 내의 개체로의 액세스를 포함할 수 있다.

<22> 여기에서 설명된 바와 같이, 개체는 호출이 개체 상에 배치될 때 실행될 수 있는 메소드를 포함할 수 있는 컴퓨터 소프트웨어 컴포넌트이다. 개체는 종래의 컴퓨터 시스템에서와 같이 생성될 수 있다. 도시된 실시예에서, 개체의 인스턴스는 개체가 스크립트(114)와 같은 실행 코드에서 인스턴스화될 때 개체 정의로부터 생성된다. 도 1의 종래기술의 예에서, 일단 개체의 인스턴스가 실행 코드에 이용가능하게 되면, 그 코드는 그 개체 상에 호출을 배치함으로써 그 개체 내의 메소드를 액세스할 수 있다.

<23> 개체 정의는 여러 방식 중의 임의의 방식으로 제공될 수 있다. 개체 정의는 컴퓨터 시스템(110)의 운영 체제의 일부로서 제공될 수 있다. 대안적으로 또는 부수적으로, 개체 정의는 특별히 스크립트(114)가 실행될 수 있게 하기 위해 컴퓨터 시스템(110) 상으로 로드될 수 있고, 또는 호스트(112)와 관련하여 제공될 수 있다. 다른 예로서, 개체 정의는 스크립트(112)의 실행에 관련되지 않은 애플리케이션 프로그램의 일부로서 로드될 수 있다.

<24> 여기에서, 컴퓨터 시스템(110) 내의 개체 정의의 컬렉션은 개체 라이브러리(116)로서 도시된다. 개체 라이브러리(116)는 여기에서 2개의 개체 정의(162 및 164)를 갖는 것으로 도시된다. 그러나, 개체 라이브러리(116)는 더 많은 개체를 위한 정의를 포함할 수 있는데, 단순화를 위해 단지 2개의 개체 정의만이 도시된다. 더욱이, 도 1은 단일 개체 라이브러리(116)를 도시하고 있다. 본 발명의 구현시에, 모든 개체 정의가 단일 파일 또는 그외 다른 위치에 정의되어야 하는 것은 아니다. 개체 라이브러리(116)는 호스트(112) 내에서 실행되는 스크립트(114)에 의한 액세스를 위해 이용가능한 개체들의 컬렉션의 논리적 표현으로서 간주될 수 있다.

<25> 개체 라이브러리(116)는 컴퓨터(110) 상에서의 스크립트의 실행시에 사용된 다수의 기능을 실행하는 개체의 정의를 포함할 수 있다. 개체 라이브러리(116) 내의 개체에 의해 실행된 기능은 시스템 자원(118)으로의 액세스를 포함할 수 있다.

<26> 시스템 자원(118)은 컴퓨터 시스템(110)의 하드웨어 또는 기타 소프트웨어 컴포넌트와 상호작용하는 컴포넌트를 포함할 수 있다. 예를 들어, 시스템 자원(152_1)은 파일 관리 시스템을 구현할 수 있다. 개체 라이브러리(116) 내에 정의된 개체는 정보를 저장하거나 검색하기 위해 파일 관리 시스템을 액세스하는 메소드를 포함할 수 있다. 그외 다른 시스템 자원은 그외 다른 기능을 실행할 수 있다. 예를 들어, 시스템 자원(152_2 또는 152_3)은 컴퓨터 시스템(110)의 인간 사용자와의 상호작용 또는 이메일 관리에 관한 기능을 실행할 수 있다. 단순화를 위해 3개의 시스템 자원(152_1...152_3)만이 도시되지만, 컴퓨터 시스템은 다수의 시스템 자원을 가질 수 있다.

<27> 각각의 시스템 자원은 관련된 개체 라이브러리(116) 내에 하나 이상의 개체 정의를 가질 수 있다. 이들 개체 정의는 시스템 자원의 기능을 액세스하기 위해 호출될 수 있는 메소드를 포함할 수 있다.

<28> 도 1은 ActiveX® 개체인 예시적인 개체를 제공한다. 따라서, 개체는 ActiveX® 개체에서 통상적인 인터페이스를 구현하는 것으로 도시된다. 그러나, 임의의 적합한 유형의 개체 및 임의의 적합한 형태의 인터페이스가 사용될 수 있다.

<29> 도 1은 ActiveX® 개체를 위해 정의된 상이한 인터페이스를 구현하는 개체 라이브러리(116) 내의 개체 정의(162 및 164)를 도시하고 있다. 이 예에서, 각 개체(162 및 164)는 인터페이스 IDispatch 및 IUnknown을 구현한다. 인터페이스 IDispatch의 구현은 스크립트(114)와 같은 스크립트 내의 명령어를 해석하는 호스트(112)로부터 개체가 호출될 수 있게 한다. IUnknown 인터페이스는 COM 표준에 따라 컴파일된 코드로부터 개체가 액세스될 수

있게 한다.

<30> 개체(162)는 IObjectSafety로서 식별된 인터페이스를 더 구현한다. IObjectSafety 인터페이스는 개체(162)의 개발자가 개체에 동작의 "안전 모드"를 제공했기 때문에 개체(162)에서 구현된다. 동작의 안전 모드 개체에서 호출될 수 있는 개체의 임의의 메소드는 보안 구역에서 실행하기에 "안전한" 것으로 간주된다. 이 예에서, 메소드는 파일을 변경하거나, 컴퓨터로부터 정보를 부적절하게 누설하거나, 그렇지 않으면 컴퓨터 시스템 또는 그 사용자에게 유해한 동작을 실행함으로써, 컴퓨터 시스템(110)의 동작을 중단시키기 위해 사용될 수 없는 경우에, "안전한" 것으로 간주된다. 하지만, 안전 모드는 임의의 적합한 방식으로 구현될 수 있다. 예를 들어, 개체는 안전 모드에서 동작하는 경우에, 시스템 자원(118)을 액세스하는 임의의 메소드가 호출될 때 사용자 통지를 생성하도록 구현될 수 있다. 개체는 그 통지 응답시에 사용자 허가를 수신하지 않으면 임의의 그러한 메소드 호출을 실행할 수 없다.

<31> 이와 대조적으로, 개체(164)는 인터페이스 IObjectSafety를 구현하지 않는다. 개체(162와 164) 사이의 이 차이는 개체가 호스트(112) 내에서 실행되는 스크립트(114)로부터 액세스될 수 있는 방식에 영향을 주어, 호스트(112)가 "보안 구역"을 구현할 수 있게 한다. 도 1의 예에서, 개체 정의(162)로부터 인스턴스화된 개체는 보안 구역으로부터 액세스될 수 있지만, 개체 정의(164)로부터 인스턴스화된 개체는 액세스될 수 없다.

<32> 여기에서 사용된 바와 같이, "보안 구역"이라는 용어는 컴퓨터 시스템(110)내의 다른 소프트웨어 컴포넌트로의 액세스를 제한하고자 하는 컴퓨터 실행가능 명령어들의 컬렉션을 설명하는 것이다. 많은 경우에, 보안 구역은 인터넷으로부터 다운로드된 스크립트를 위해 구현되지만, 임의의 신뢰할 수 없는 코드를 위해 구현될 수 있고, 또는 소정의 소프트웨어가 컴퓨터 시스템의 다른 컴포넌트로의 제한된 액세스를 갖는 것이 요구되는 임의의 다른 상황에서 구현될 수 있다.

<33> 스크립트(114)는 개체(162)를 인스턴스화하는 컴퓨터 실행가능 명령어를 포함할 수 있다. 그러한 명령어 해석시에, 호스트(112)는 개체(162)의 인스턴스를 생성하기 위해 개체 라이브러리(116)를 액세스한다. 보안 구역으로부터 생성된 경우, 개체(162)의 인스턴스는 "안전 모드"에서 동작할 것이다. 그 후, 그 개체 상에서의 임의의 호출은 개체의 인스턴스를 통해 이루어진다. 이러한 방식으로, 보안 구역에서 실행되는 스크립트(114)는 보안 구역에서 안전하다고 간주된 개체 정의(162)의 그러한 메소드들로의 액세스를 갖는다.

<34> 이와 대조적으로, 개체 정의(164)는 인터페이스 IObjectSafety를 포함하지 않기 때문에, 스크립트(114)가 그 개체 정의로부터 인스턴스를 생성하는 명령어를 포함하더라도, 호스트(112)에 의해 인스턴스화되지 않는다. 개체(164)의 인스턴스가 형성되어야 한다는 것을 나타내는 스크립트(114) 내의 임의의 명령어는 호스트(112)가 예외를 생성하게 한다. 그러한 예외에 대한 응답시의 처리는 스크립트(114)의 실행 종료 또는 다른 적절한 에러 보정 단계를 초래할 수 있다. 그러나, 개체(164)의 어떤 인스턴스화도 생성되지 않기 때문에, 개체(164) 상의 메소드를 호출하는 스크립트(114) 내의 명령어는 실행되지 않아야 한다. 이러한 방식으로, 안전하지 않은 것으로 간주된 개체(164)의 메소드는 스크립트(114), 또는 보안 구역 내에서부터 실행되는 다른 프로그램으로 액세스 불가능하다.

<35> 도 1은 또한, 그럼에도 불구하고-보안 구역임에도 불구하고- 스트립트(114)가 안전하지 않다고 간주되는 기능을 실행할 수 있는 개체로의 액세스를 얻을 수 있는 방법을 나타낸 것이다. 보안 구역은 스크립트(114)가 개체 정의(164)로부터 개체를 직접 인스턴스화하지 못하게 하지만, 그러한 개체는 개체(162)의 메소드로부터 인스턴스화될 수 있다. 예를 들어, 스크립트(114)는 스크립트(114)로부터 다음에 액세스될 수 있는 개체(162)를 인스턴스화할 수 있다. 개체(162) 상에서의 메소드 호출은 개체(164)가 인스턴스화되게 할 수 있다. 개체(162) 상에서 호출된 메소드는 개체(164)의 인스턴스화를 스크립트(114)로 반환할 수 있다. 이러한 방식으로, 호스트(112)에 의해 구현된 보안 구역이 개체(164)로의 직접 액세스를 금지했을 수 있지만, 스크립트(114)는 개체(164)의 인스턴스화로의 액세스를 얻는다. 그 다음, 스크립트(114)는 시스템 자원(118)을 액세스하는 개체(164) 상의 메소드를 호출할 수 있고, 부주의로 또는 고의적으로 컴퓨터 시스템(110)의 동작을 중단시킬 수 있다.

<36> 도 2는 본 발명의 실시예에 따라 컴퓨터 시스템을 동작시키는 프로세스를 도시한 것이다. 프로세스는 이 간접 인스턴스화를 통해 안전하지 않다고 간주된 메소드에 스크립트 액세스를 제공하지 못하게 한다. 도 2의 프로세스는 개체의 액세스 제어가 각 메소드 호출마다 제공될 수 있게 한다. 개체가 보안 구역으로부터 직접 또는 간접적으로 인스턴스화되었는 지에 상관없이, 안전하지 않다고 간주된 메소드로의 호출이 그 개체 상에 배치되면, 호출은 거부되거나, 그렇지 않으면 안전하지 않다고 간주된 메소드로의 액세스를 제한하도록 처리될 수 있다.

<37> 도 2의 프로세스 단계는 컴퓨터 시스템 내의 하나 이상의 소프트웨어 컴포넌트에 의해 실행될 수 있다. 몇몇 실시예에서, 도 2의 프로세스는 스크립트 내의 명령어를 해석하는 웹 브라우저 내의 컴포넌트에 의해 실행된다.

<38> 도 2의 프로세스는 스크립트 내의 명령어를 해석하는 종래의 단계에서 시작될 수 있다. 단계(210)에서, 프로세스는 개체를 인스턴스화하는 명령어에서부터 시작된다. 그러한 명령어는 스크립트 내에 인코드될 수 있다.

<39> 프로세스는 판정 블록(212)으로 진행된다. 판정 블록(212)에서, 개체가 보안 구역 내에서 인스턴스화하기에 안전한 지의 여부에 관해 판정이 이루어진다. 이 예에서, 개체는 IObjectSafety와 같은 제한적인 인터페이스로 정의되면 "안전한" 것으로 간주된다. 그러나, 개체가 안전한지 판정하기 위해 임의의 적합한 기준이 이용될 수 있다.

<40> 개체가 안전하다고 간주되지 않으면, 프로세스는 종료점(214)으로 진행된다. 종료점(214)은 예외의 생성, 에러 핸들러의 실행, 또는 보안 구역에서 실행하기에 안전하다고 간주되지 않은 개체를 인스턴스화하려는 스크립트에 의한 시도에 응답하는 다른 단계의 실행을 수반할 수 있다. 보안 구역에서 실행하기에 안전하다고 간주되지 않은 개체를 인스턴스화하려는 시도의 처리는 종래의 컴퓨터 시스템에서처럼 될 수 있지만, 임의의 적합한 처리가 실행될 수 있다.

<41> 이와 대조적으로, 개체가 안전하다고 간주되면, 처리는 블록(216)으로 진행된다. 블록(216)에서, 개체는 인스턴스화될 수 있다. 개체의 인스턴스화는 종래의 컴퓨터 시스템에서처럼 실행될 수 있지만, 임의의 적합한 메커니즘이 개체를 인스턴스화하기 위해 사용될 수 있다. 종래의 컴퓨터 시스템에서처럼, 보안 구역으로부터의 개체의 인스턴스화는 개체가 안전 모드에서 실행되고 있다는 것을 기록하는 것을 포함할 수 있다. 개체의 인스턴스가 안전 모드에서 동작하고 있다는 것의 기록은 임의의 적합한 방식으로 달성될 수 있지만, 다음 예에서; 개체의 각각의 인스턴스화는 여기에서 "SafeMode"로 칭해지는 부울 변수의 값을 저장하기 위한 메모리로의 액세스를 갖는다. SafeMode의 값은 인스턴스가 안전 모드에서 실행하도록 생성될 때 참(true)으로 설정된다.

<42> 그 다음, 호스트는 스크립트 내의 또 다른 명령어를 실행할 수 있다. 메소드에 대한 호출을 수신하면, 처리는 블록(218)으로 계속된다. 몇몇 실시예에서, 메소드에 대한 호출은 스크립트를 처리하는 동안 이루어진 모든 메소드 호출을 인터셉트하는 컴포넌트에 의해 수신되어 처리된다. 스크립트를 해석하는 종래의 시스템에서처럼, 메소드 호출을 지정하는 명령어는 때때로 스크립트 엔진이라 불리는 호스트 내의 소프트웨어 컴포넌트를 호출함으로써 처리된다. 스크립트 엔진은 개체를 인출하고, 그 개체에 대한 인터페이스를 획득할 수 있다.

<43> 종종, 소프트웨어 시스템 내의 다수의 개체에 대한 인터페이스는 공통 처리 단계를 실행하는 공유 코드로 구현될 수 있다. 인터페이스를 구현하기 위한 공유 코드는 여기에서 "호출자(invoker)"라 불린다.

<44> 종래 기술의 컴퓨터 시스템에서, 호출자는 하나 이상의 인터페이스를 구현하고, 호출자에 의해 구현된 인터페이스에 적절한 포맷의 호출 명령어를 입력으로서 수신한다. 호출자는 메소드를 식별하는 인수 및 그 메소드를 호출할 때 패스될 인수를 포함하는 호출을 수신한다. 호출자는 이 입력 정보를 구문 분석하고, 원하는 메소드를 실행하는 컴퓨터 실행가능 코드가 저장되는 컴퓨터 시스템(110) 내의 특정 위치를 판정한다. 그 다음, 호출자는 이 코드를 액세스하고, 결과를 수신하여, 이 결과를 포맷해서, 호출자에 의해 구현된 인터페이스를 통해 이 결과를 호출 프로그램에 반환한다.

<45> 도 2에 도시된 본 발명의 실시예에 따른 호출자는 종래의 시스템에서처럼 호출자의 기능들 중의 임의의 기능을 실행할 수 있다. 그러나, 도 2의 프로세스에서 사용된 호출자는 수신된 메소드 호출에 기초하여 처리를 실행하도록 구성될 수 있다. 보안 구역이 구현되는 도 2의 예에서, 메소드 호출의 처리는 호출이 허용되어야 하는지, 거부되어야 하는지 그렇지 않으면 변경되어야 하는지의 여부를 판정할 수 있다. 그러므로, 각 메소드 호출은 호출자에 의해 조건부로 처리된다. 또한, 메소드 호출이 허용될 때, 호출자는 메소드를 호출하기 전에 메소드로의 입력을 처리할 수 있거나, 또는 호출 프로그램에 결과를 반환하기 이전에 메소드에 의해 생성된 결과를 처리할 수 있다.

<46> 도 2의 프로세스의 나머지 단계는 호출자에 의해 실행될 수 있는 조건부 처리의 예를 제공한다. 판정 블록(220)에서, 호출이 보안 구역으로부터 이루어졌는지 판정이 이루어진다. 몇몇 실시예에서, 메소드 호출은 메소드 호출이 배치되는 개체가 참으로 설정된 SafeMode 필드를 가지면 보안 구역으로부터 이루어진 것으로 간주된다. 상기 설명된 바와 같이, 보안 구역에서 직접 또는 간접적으로 인스턴스화된 모든 개체는 참으로 설정된 SafeMode 필드를 갖는다. 그러나, 호출이 보안 구역으로부터 이루어졌는지 여부의 판정은 임의의 적합한 방식으로 이루어질 수 있다.

<47> 메소드 호출이 보안 구역으로부터 이루어지지 않으면, 처리는 블록(222)으로 진행된다. 블록(222)에서, 메소드는 종래의 처리를 사용하여 호출이 배치될 수 있게 함으로써 실행된다. 블록(224)에서, 메소드 실행 결과가 호출 프로그램으로 반환되고, 프로세스가 종료된다. 결과는 호출 프로그램으로의 반환을 위해 종래의 호출 처리에서와 같이 포맷될 수 있다.

<48> 이와 반대로, 호출이 보안 구역으로부터 이루어지면, 호출이 조건부로 실행되거나 달리 처리될 수 있는 서브 프로세스(226)가 실행된다. 서브 프로세스(226)는 블록(230)에서 시작된다. 블록(230)에서, 호출된 메소드가 보안 구역에서 실행하도록 허가되었는지 판정이 이루어진다. 메소드가 보안 구역에서의 실행을 위해 허가되었는지 판정하는 임의의 적합한 메커니즘이 사용될 수 있다. 몇몇 실시예에서, 개체의 개발자는 전체 개체가 보안 구역으로부터 실행하기에 안전한지 판정하기 위해 통상적으로 적용되는 동일한 기준을 사용하여 보안 구역에서 실행하기에 적합한 메소드를 식별할 것이다. 그러나, 블록(230)에서의 판정은 전체 개체에서보다는 오히려 각 메소드마다 제공된다.

<49> 그 다음, 처리는 판정 블록(232)으로 진행된다. 판정 블록(232)에서, 프로세스는 호출된 메소드가 실행을 위해 허가되었는 지의 여부에 기초하여 분기된다. 호출된 메소드가 허가되지 않았으면, 처리는 에러 종료(234)로 진행된다. 임의의 적합한 처리가 에러 종료(234)에서 실행될 수 있다. 예를 들어, 에러 종료(234)에서의 처리는 예외의 생성, 에러 핸들러의 실행 또는 다른 적합한 응답을 초래할 수 있다.

<50> 이와 반대로, 호출된 메소드가 보안 구역에서의 실행하도록 허가되면, 처리는 메소드로의 입력이 메소드로의 호출 이전에 처리되는 블록(235)으로 진행할 수 있다. 임의의 원하는 처리가 입력상에서 실행될 수 있지만, 몇몇 실시예에서는 입력상에서 어떤 처리도 실행되지 않는다. 예를 들어, 그러한 처리는 성능 모니터링 애플리케이션의 일부로서 입력 값을 기록하기 위해 사용될 수 있다.

<51> 블록(235)에서 입력이 처리되는 지의 여부에 상관없이, 처리는 그 다음에 블록(236)으로 진행된다. 블록(236)에서, 메소드가 실행된다. 메소드는 블록(222)에서 실행된 것과 동일한 방식으로 블록(236)에서 실행될 수 있다.

<52> 그러나, 블록(236)에서의 메소드 실행으로부터 얻은 결과는 더욱 처리될 수 있다. 도시된 실시예에서, 판정 블록(240)은 결과가 개체인지 판정하기 위해 결과를 더욱 처리한다. 상이한 유형의 결과를 처리하는 컴퓨터 시스템은 공지되어 있고, 그러한 시스템은 결과의 유형에 의존하여 다르게 결과를 처리할 수 있다. 종래의 기술은 결과가 개체인지 식별하기 위해 이용될 수 있다. 예를 들어, 결과가 특정 포맷의 포인터이면 개체가 식별될 수 있다.

<53> 결과가 개체인지 식별하기 위해 사용된 특정 메커니즘에 상관없이, 결과가 개체가 아니면, 처리는 블록(242)으로 진행된다. 블록(242)에서, 결과는 호출 프로그램으로 반환된다. 결과는 종래의 형태로 블록(242)에서 반환될 수 있다. 그러므로, 블록(242)에서의 처리는 블록(224)에서 실행된 처리와 동일할 수 있다.

<54> 그러나, 결과가 개체이면, 처리는 판정 블록(242)에서 블록(250)으로 진행된다. 블록(250)에서, 래퍼는 호출된 메소드의 실행 결과로서 생성되는 개체에 대해 생성된다. 여기에서, "래퍼"라는 용어는 래핑된 소프트웨어 컴포넌트와 그것을 호출할 수 있는 외부 컴포넌트 사이에 삽입되는 소프트웨어 컴포넌트를 칭하는 것이다. 래퍼는 래핑된 컴포넌트와 동일한 형태로 외부 컴포넌트 인터페이스에 나타난다. 래퍼는 그 다음에, 미리 정의된 인터페이스를 통해 래핑된 컴포넌트를 액세스한다. 그러나, 래퍼는 래핑된 컴포넌트 상에 호출을 배치하기 이전에 외부 컴포넌트로부터의 호출에 관한 처리를 실행할 수 있고, 또는 래핑된 컴포넌트에 의해 제공된 결과를 외부 컴포넌트에 제공하기 전에 그 결과를 처리할 수 있다.

<55> 이 예에서, 블록(250)에서 생성된 래퍼는 개체가 안전 모드로 생성되었다는 표시를 기록하기 위한 저장 위치를 포함한다. 블록(252)에서, 이 SafeMode 필드는 참으로 설정된다.

<56> 처리는 그 다음에 블록(254)으로 진행된다. 블록(254)에서, 래퍼는 호출된 메소드의 실행 결과로서 반환된다. 개체가 호출된 메소드의 실행 결과로서 인스턴스화되었을 수 있지만, 호출 프로그램은 개체가 직접 액세스될 수 있게 할 수 있는 개체에 대한 정보를 수신하지 않는다. 오히려, 호출 프로그램은 인스턴스화된 개체에 대한 래퍼에 관한 정보를 수신할 뿐이다. 결과적으로, 그 개체 상에서 이루어진 모든 후속 호출은 블록(250)에서 생성된 래퍼를 통해 이루어질 것이다.

<57> 인스턴스화된 개체가 보안 구역에서 생성되었다는 어떤 표시도 포함하지 않을지라도, 래퍼는 그러한 표시를 포함할 것이다. 결과적으로, 그 개체 내의 메소드 상의 임의의 후속 호출은 안전 모드 서브 프로세스(226)에 따

라 처리될 것이다. SafeMode 변수를 저장하기 위한 래퍼를 생성함으로써, 보안 구역 처리가 요구된다는 표시는 개체들이 보안 구역에서 간접적으로 생성되더라도, 한 개체에서 다른 개체로 전달된다. 그 개체에 의해 인스턴스화된 임의의 또 다른 개체는 마찬가지로 보안 구역에서 처리하기 위해 생성된 것으로 표시될 것이다.

<58> 도 3을 참조하면, 도 2의 프로세스를 실행할 수 있는 컴퓨터 시스템의 소프트웨어 아키텍처가 도시된다. 도 3은 호스트(312)를 도시하고 있다. 예를 들어, 호스트(312)는 인터넷으로부터 다운로드된 스크립트(314)가 실행되는 보안 구역을 실현하는 웹 브라우저일 수 있다. 이와 관련하여, 스크립트(314)는 신뢰할 수 없는 코드에 의해 실행하기에 안전하지 않은 것으로 간주되는 개체로의 액세스가 제공되어서는 안 되는 신뢰할 수 없는 코드를 나타낼 수 있다.

<59> 그러나, 대안적인 실시예가 가능하다. 예를 들어, 호스트(312)는 서명된 C++ 코드와 같은 신뢰 코드를 실행할 수 있다. 이 실시예에서, 메소드 호출의 처리는 시스템 자원을 조작 처리하는 개체를 신뢰 코드만이 액세스할 수 있게 실행될 수 있다. 대안적으로, 이 실시예에서의 처리는 신뢰 코드가 신뢰할 수 없는 코드를 액세스하지 않는다는 것을 보장하도록 실행될 수 있다.

<60> 스크립트(314)는 여기에서, 호스트(312)에 의해 해석되는 다수의 명령어를 포함하는 것으로 도시된다. 도 3에 도시된 명령어는 임의의 특정 프로그래밍 언어를 나타내지 않은 단순화된 형태로 되어 있다. 스크립트(314)는 설명된 기능을 실행하기 위해 임의의 적합한 언어로 된 명령어로 구현될 수 있다.

<61> 명령어(316)는 개체 A로서 식별된 개체를 인스턴스화하기 위한 명령어이다. 명령어(316)의 실행은 개체 A의 인스턴스(362)가 생성되게 한다. 인스턴스(362)는 종래의 컴퓨터 시스템에서와 같이 개체 정의로부터 생성될 수 있다. 이 예에서, 컴퓨터 판독가능 매체는 개체 정의를 포함하는 클래스(340)를 저장한다. 도 3에서, 개체 정의(342 및 344)가 도시된다. 단지 2개의 그러한 개체 정의만이 단순화를 위해 도시되지만, 컴퓨터 시스템은 다수의 개체 정의를 각각 포함하는 다수의 클래스를 포함할 수 있다.

<62> 명령어(316)의 실행시에, 호스트(312)는 개체 A의 인스턴스를 생성하기 적절한 개체 정의를 얻기 위해 클래스(340)를 액세스한다. 명령어(316)는 예시를 위해 단순화된 구문으로 작성된다. 개체의 인스턴스화를 지정하는 실제 명령어는 도 3에 도시된 것보다 더 많은 정보를 필요로 할 수 있는 개체의 인스턴스 생성시에 사용하기에 적절한 개체 정의를 지정하기 위해 충분한 정보를 포함할 것이다. 이 예에서, 개체 정의(342)는 개체의 인스턴스를 생성하기 위해 사용된다.

<63> 일단 개체의 인스턴스가 생성되면, 그 인스턴스는 호스트(312)에 의해 생성된 보안 구역 내에서 실행되는 명령어에 응답하여 인스턴스가 생성되었기 때문에, 안전 모드에서 동작하도록 설정될 수 있다. 이 예에서, 인스턴스(362) 내의 SafeMode 필드(372)는 참으로 설정된다.

<64> 명령어(318)는 명령어(316)에서 생성된 개체의 인스턴스 상의 메소드(여기에서 method1로 표시됨)를 호출하는 명령어이다. 호스트(312)는 종래 시스템에서의 호출 명령어의 처리와 유사한 방식으로 명령어(318)를 처리할 수 있다. 종래의 시스템에서, 명령어(318)와 같은 명령어는 IDispatch 인터페이스를 구현하는 컴포넌트를 통해 처리된다. 따라서, 도 3은 명령어(318)가 IDispatch 인터페이스를 통한 개체의 액세스의 일부로서 호출자(330)에 의해 초기에 처리되도록, IDispatch 인터페이스를 구현하는 호출자(330)를 도시하고 있다.

<65> 호출자(330)는 종래 시스템에서의 호출자와 다른데, 그 이유는 그 호출자가 명령어(318)에서 식별된 메소드에 조건부로 호출을 배치하기 때문이다. 도시된 실시예에서, 호출자(330)는 호출이 허용되어야 하는지 판정하기 위해 method1에 호출을 배치하기 이전에 호출 명령어(318)를 처리한다.

<66> 호출자(330)는 메소드가 SafeMode에서 실행하도록 허가되는지 판정하기 위해 호출 명령어(318)를 처리한다. 도시된 실시예에서, 호출자(330)는 호출된 메소드를 포함하는 개체를 위한 개체 정의와 관련된 메타데이터로부터의 SafeMode에서 실행하도록 호출된 메소드가 허가되는지 판정한다. 도 3에 도시된 예에서, 메타데이터는 화이트 리스트(white list)(350)로서 클래스(340) 내에 저장된다. 화이트 리스트(350)는 SafeMode에서 실행하도록 허가되는 클래스(340) 내에서 정의되는 메소드 리스트를 포함한다. 화이트 리스트(350) 상에서 method1을 찾으면, 호출자(330)는 method1이 실행되도록 개체 A의 인스턴스(362) 상에서의 호출을 허용할 수 있다.

<67> 도 3은 클래스(340)에서 정의된 모든 메소드에 대한 정보를 포함하는 클래스(340)의 단일 화이트 리스트(350)를 도시하고 있다. 화이트 리스트는 각 클래스마다 제공될 수 있다. 그러나, 화이트 리스트는 소프트웨어 계층의 임의의 적합한 레벨과 관련될 수 있다. 예를 들어, 다수의 화이트 리스트가 각 클래스마다 제공되어, 각 개체 정의마다 하나의 화이트 리스트가 있을 수 있다. 대안적으로, 화이트 리스트는 호스트(312)와 관련되거나, 또는 컴퓨터 시스템 내의 다른 소프트웨어 컴포넌트와 관련될 수 있다. 게다가, 다수의 화이트 리스트 및/또는

메타데이터를 저장하는 다수의 데이터 구조는 메소드 호출에 관한 조건부 처리의 일부로서 제공되어 참고될 수 있다.

<68> 도시된 실시예에서, 개체 A는 ActiveX® 개체이고, 호출은 인터페이스를 통해 ActiveX® 개체 내의 실행가능 코드에 배치될 수 있다. 이 예에서, 호출은 종래의 컴퓨터 시스템에서와 같이 인터페이스 IDispatch를 통해 배치되지만, IUnknown과 같은 다른 인터페이스가 사용될 수 있다. 이 예에서, 개체가 IDispatch 인터페이스를 통해 호출될 때, 호출자(330)는 호출을 인터셉트하고, 호출된 메소드가 실행하도록 허가되면 선택적으로 호출이 진행될 수 있게 한다. 허가된 경우, 호출은 인터페이스 IDispatch를 통해 ActiveX® 개체로 종래의 호출에서처럼 진행된다. 그러나, 호출이 진행될 수 있게 하는 임의의 적합한 메소드가 사용될 수 있다.

<69> 스크립트(314)는 개체를 액세스하는 다른 명령어를 포함할 수 있는데, 이것은 유사한 방식으로 처리될 수 있다. 예를 들어, 명령어(320)는 개체 B를 인스턴스화하는 명령어이고, 명령어(322)는 개체 B 상의 메소드를 호출하는 명령어이다. 명령어(320)의 실행시에, 호스트(312)는 개체 A의 인스턴스(362)가 명령어(316)에 응답하여 생성된 것과 동일한 방식으로 개체 B의 인스턴스(364)를 생성할 수 있다. 이 예에서, 클래스(340) 내의 개체 정의(344)는 인스턴스(364)를 생성하기 위해 사용된다. 인스턴스(362)에서와 같이, 인스턴스(364)는 인스턴스(364)가 보안 구역 내에서부터 실행된 명령어에 응답하여 생성되었다는 것을 나타내는, 참으로 설정되는 SafeMode 변수를 포함한다. 이 예에서, 인스턴스(364)는 또한, 인스턴스(364)를 생성하기 위해 사용된 개체 정의(344)의 일부로서 정의되는 method2로의 엔트리를 제공한다.

<70> 명령어(322)가 실행될 때, 호출이 명령어(318)의 실행에 응답하여 method1 상에서 이루어진 것과 거의 동일한 방식으로 호출이 method2 상에서 이루어진다. method1 및 method2 둘 다 화이트 리스트(350) 상에 포함되기 때문에, 이 둘의 호출은 허용된다. 그러나, 개체 A 및 B의 각각은 화이트 리스트(350) 상에 열거되지 않은 메소드를 포함할 수 있다. 호스트(312) 내에서 실행된 임의의 코드로부터 그러한 메소드를 호출하려는 시도는 그 메소드에 대한 호출이 거부되게 할 것이다.

<71> 예를 들어, 개체 B는 method3의 엔트리(386)를 포함하는 것으로 도시된다. 도 3에 도시된 예에서, method3은 화이트 리스트(350) 상에 포함되지 않는다. 따라서, method3에 대한 임의의 호출은 거부될 것이다. 도 2와 관련하여 위에서 설명된 바와 같이, 허가되지 않은 메소드에 대한 호출은 임의의 적합한 방식으로 처리될 수 있다.

<72> 도 4는 본 발명에 따른 컴퓨터 시스템의 대안적인 실시예를 도시한 것이다. 도 4는 호스트(312)(도 3)와 동일한 방식으로 구현될 수 있는 호스트(412)를 도시하고 있다. 호스트(412)는 호출자(430)에 의해 처리되는 메소드 호출을 포함하는 스크립트(414)를 실행한다. 호출자(430)는 마찬가지로, 호출자(330)와 동일한 형태로 될 수 있다.

<73> 도 3의 스크립트(314)에서와 같이, 스크립트(414)는 개체 A를 인스턴스화하는 명령어(416)를 포함한다. 이 명령어에 응답하여, 개체 A의 인스턴스(462)는 클래스(440) 내의 개체 정의(442)로부터 생성된다. 명령어(416)가 보안 구역 내에서부터 실행되기 때문에, 명령어(462)는 참으로 설정된 SafeMode 변수(472)를 포함한다

<74> 실행시에, 명령어(418)는 개체 A 내의 method1을 호출한다. 명령어(418) 처리시에, 호출자(430)는 method1이 호스트(412)에서 구현된 보안 구역에서 실행하도록 허가되는 지의 여부를 판정한다. 도 4의 실시예에서, 클래스(440)는 실행하도록 허가된 메소드를 식별하는 메타데이터를 포함한다. 이 예에서, 메타데이터는 블랙 리스트(450)이다. 블랙 리스트(450)는 SafeMode에서 실행하도록 허가되지 않는 메소드의 리스트를 포함한다. 이 예에서, method1은 블랙 리스트(450) 상에 포함되지 않는다. 따라서, 명령어(418)가 실행될 때, 호출자(430)는 method1에 대한 호출이 진행될 수 있게 한다.

<75> 도 4에 도시된 시나리오에서, 개체 A 내의 method1 상에 메소드 호출을 배치하는 것은 개체 B를 간접적으로 인스턴스화한다. 이 예에서, method1은 엔트리(474)를 갖는 것으로 도시된다. 엔트리(474) 다음에, 개체 B는 명령어(476)에 의해 인스턴스화된다. 명령어(476)의 실행은 개체 B의 인스턴스(464)가 생성되게 한다. 개체 B의 인스턴스(464)는 종래의 방식으로 생성될 수 있다.

<76> 인스턴스(464)가 method1에서 실행된 명령어(476)에 응답하여 생성된 후, 개체 B는 method1의 실행 결과로서 명령어(478)에서 반환된다. 도시된 실시예에서, 개체 B는 포인터에 의해 나타내진다. 그러나, 임의의 적합한 데이터 구조가 개체를 나타내기 위해 사용될 수 있다. method1의 실행 다음에 개체를 반환하기 위해, 명령어(478)는 개체 B에 대한 포인터를 반환한다.

<77> 호출자(430)는 그 결과의 반환이 안전하지 않은 코드, 또는 소프트웨어 시스템의 다른 제한된 부분으로의 액세스를 스크립트(414)에 제공하는지 또는 제공할 수 있을지 판정하기 위해 각 결과를 처리할 수 있다. 도 4의 예에서, 개체 B는 안전하지 않다고 간주된 액션을 실행할 수 있는 메소드를 포함할 수 있다. 호출자(430) 내에서의 처리는 메소드들 중의 어느 것이 "안전하지 않은지" 판정하기 위해 개체 B의 각 메소드를 분석할 수 있다. 하지만, 반환된 개체가 안전하지 않은 메소드를 포함할 가능성이 있기 때문에, 반환된 임의의 개체가 "안전하지 않다"고 간주될 수 있다. 이와 유사한 처리는 안전하지 않을 수 있는 어레이 또는 다른 유형의 변수에서 실행될 수 있다.

<78> 안전하지 않은 결과를 만들어내는 반환 변수를 식별하기 위해 호출자(430)에 의해 사용된 특정 메커니즘에 상관없이, 호출자(430)는 또한 스크립트(414)가 그 결과로의 직접적이고 보호되지 않은 액세스를 확실히 얻지 않게 하기 위한 단계를 취할 수 있다. 설명된 실시예에서, 래퍼는 안전하지 않다고 간주된 반환 변수로의 액세스를 제어하기 위해 사용된다.

<79> 인스턴스(464)가 안전하지 않다고 간주되기 때문에, 호출자(430)는 인스턴스(464)의 래퍼(480)를 생성한다. 래퍼(480)는 개체 B에 인터페이스를 제공하는 컴퓨터 실행가능 명령어들의 그룹을 나타낸다. 래퍼(480)는 래퍼(480)가 SafeMode에서 생성되었다는 것을 나타내는 한 비트를 저장하는 필드(482)를 포함한다. 필드(482)는 래퍼(480)가 생성될 때 참으로 설정된다. 그 다음, 호출자(430)는 개체 B를 반환하기보다는 오히려 호출 프로그램에 래퍼(480)를 반환한다.

<80> 스크립트(414) 내의 명령어(418)의 실행 다음에, 스크립트(414)는 개체 B를 액세스하는데 필요한 정보를 갖는다. 따라서, 개체 B 내의 method2를 호출하는 명령어(420)의 실행시에, 호출자(430)는 method2로의 호출을 처리할 수 있다. 하지만, 도시된 실시예에서, 개체 B로의 액세스는 래퍼(480)를 통한다.

<81> 도 4에 도시된 바와 같이, 래퍼(480)는 인터페이스를 제공하는데, 이 인터페이스를 통해, 호출자(430)는 개체 B 상에 배치된 메소드 호출에 응답하여 액세스될 수 있다. 래퍼(480)에 의해 제시된 인터페이스는 보안 구역에서 인스턴스화된 다른 개체에 의해 제시된 인터페이스와 동일한 형태로 이루어질 수 있다. 그러므로, 래퍼(480)를 통해 개체 B 상에 호출이 배치될 때, 호출자(430)는 다른 개체 상에 배치된 메소드 호출을 처리하는 것과 동일한 방식으로 그들 호출을 처리할 수 있다. 래퍼(480)를 통한 개체 B 상의 호출 배치는 그 호출이 보안 구역에서 허가되는지 확인될 수 있게 한다. 이 예에서, 래퍼(480)는 참으로 설정된 SafeMode 변수(482)를 갖는데, 이것은 래퍼(480)를 통해 이루어진 임의의 메소드 호출이 제한되는 지의 여부를 호출자(430)가 확인하게 한다. 그러므로, 래퍼(480)를 통한 호출은 안전 모드 실행에 적절하게 처리된다. 개체 B 내의 호출된 메소드가 보안 구역 내에서의 액세스를 위해 허가되면, 호출자(430)는 그러한 호출의 처리시에, 호출된 메소드를 실행하는 개체 B 내의 코드를 액세스할 것이다. 도 4의 예에서, method2는 블랙 리스트(450) 상에 열거된다. 따라서, 호출자(430)가 명령어(420)를 처리할 때, 호출자(430)는 method2에 대한 호출을 거부한다. 예외 생성 또는 에러 핸들러의 실행을 포함하여 임의의 적합한 응답이 메소드 호출의 거부시에 취해질 수 있다.

<82> 도시된 예에서, 개체 B가 간접적으로 인스턴스화되었지만, 도 4의 실시예에 의해 실행된 각 메소드 호출 마다의 허가는 method2가 보안 구역에서 실행하도록 허가되지 않았기 때문에 method2가 실행되지 않게 한다. 이러한 방식으로, 액세스 제어는 보안 구역에서 간접적으로 인스턴스화된 개체에 대해서도 구현된다.

<83> 더욱이, 호출자(430)와의 인터페이스를 구현하기 위해 개체를 요구하는 것 이외에는, 도 4의 실시예에서 예시된 각 메소드 호출 마다의 액세스를 제공하기 위해 개체 B의 정의를 생성하기 위한 어떤 특별한 코딩도 요구되지 않았다. 화이트 리스트(350), 블랙 리스트(450) 또는 다른 메타데이터로의 액세스는 각 개체 내에 어떤 코딩도 추가되지 않고 호출자 내에서 발생한다. 스크립트(414)로부터 실행된 개체의 특별한 코딩의 필요성을 없앰으로써, 스크립트가 실행될 수 있는 컴퓨터 시스템에서 사용하기 위한 개체를 생성하는 개발자의 부담은 감소된다. 더욱이, 컴퓨터 시스템(110)(도 1)과 같은 컴퓨터 시스템은 스크립트(414)를 실행하는 호스트(412) 이외에 많은 유형의 프로그램에 의해 액세스되는 개체를 포함할 수 있다. 각 메소드 호출마다 액세스 제어를 구현하기 위한 개발자의 부담 감소는 호스트(412) 내에서 실행되는 스크립트가 다른 프로그램에 의해 컴퓨터 시스템 내에 설치된 개체를 이용할 수 있는 기회를 증가시킨다.

<84> 본 발명의 최소한 한 실시예의 몇몇 실시양상을 이렇게 설명했으므로, 다양한 변경, 변형 및 구현이 본 분야에 숙련된 기술자들에게 용이하게 생각날 것이라는 것을 알 수 있을 것이다.

<85> 예를 들어, 본 발명은 보안 구역을 구현하는 웹 브라우저에 의해 예시된다. 그러한 실시예에서, 각 메소드 호출은 실행하기 안전한지 판정하기 위해 처리된다. 그러나, 메소드 호출을 인터셉트하여 처리하는 설명된 방법

은 다른 상황에서 적용될 수 있다. 예를 들어, 스크립트를 실행하는 임의의 프로그램은 메소드 호출에 관한 처리가 실행될 수 있게 메소드 호출을 인터셉트하도록 동작될 수 있다. 한 예로서, 메소드 호출을 로그하는 프로그램은 메소드 호출을 인터셉트할 수 있다. 다른 예로서, 메소드 호출을 인터셉트하는 프로그램은 프로그램의 실행 비용을 계산할 수 있고, 또는 메소드 호출을 다르게 처리할 수 있다. 더욱이, 메소드 호출은 화이트 리스트, 블랙 리스트 또는 다른 적합한 수단에 따라 선택적으로 처리될 수 있다.

<86> 더욱이, 본 발명은 스크립트의 처리에 제한되지 않는다. 본 발명은 개체로의 선택적인 액세스가 요구되는 임의의 상황에서 이용될 수 있다.

<87> 또한, 제한된 인터페이스를 구현하는 개체는 IObjectSafety로서 공지된 인터페이스를 구현하는 것으로 설명된다. 이 인터페이스는 ActiveX® 개체가 사용될 때 이용가능한 예로서 쓰인다. 보안 인터페이스는 임의의 적합한 방식으로 구현될 수 있다. 한 예로서, 단일 인터페이스는 보안 구역 내에서 동작하고 있는지에 상관없이 개체 상에 호출을 배치하기 위해 제공될 수 있지만, 그 인터페이스는 동작 모드를 식별하기 위해 설정될 수 있는 파라미터를 포함할 수 있다.

<88> 또한, 호출자(430)는 인터페이스 IDispatch를 통해 이루어진 모든 호출을 인터셉트한다는 것이 설명된다. 호출자(430)는 추가적인 또는 기타 인터페이스를 통해 이루어진 호출을 인터셉트할 수 있다. 예를 들어, 호출자(430)는 인터페이스 IUnknown 또는 IDispatchEx, 또는 임의의 다른 적합한 인터페이스를 통해 호출을 인터셉트할 수 있다.

<89> 또한, 래퍼는 메소드 호출이 개체를 반환할 때 생성된다는 것이 설명되었다. 다른 반환 변수는 마찬가지로 보안 염려를 일으킬 수 있고, 래퍼 내에 반환될 수 있다. 예를 들어, 비보호 개체는 어레이 내의 엔트리로서 반환될 수 있다. 호출자(330 또는 430)는 예를 들어, 메소드 호출의 결과로서 반환될 어레이 내의 각 엔트리를 분석할 수 있다. 비보호 개체를 포함하는 임의의 엔트리는 그들 개체에 대한 래퍼로 대체될 수 있다. 대안적으로, 호출자는 단순히, 어레이를 포함하는 임의의 반환 값을 인정하지 않을 수 있다.

<90> 이와 유사하게, 메소드는 호출 프로그램이 비보호 개체 또는 다른 안전하지 않은 코드로 액세스할 수 있게 하는 방식으로 입력 변수를 변경할 수 있다. 호출자(330 또는 430)는 비보호 개체로 액세스를 허용하도록 변경될 수 있는 어레이를 포함하는 것들을 식별하기 위해 입력 변수를 스캔할 수 있다. 그러한 입력 변수를 갖는 호출은 거부될 수 있고, 또는 안전하지 않은 개체로 액세스를 제공하지 않도록 다르게 처리될 수 있다.

<91> 그러한 변경, 변형 및 구현은 이 명세서의 일부가 될 수 있고, 본 발명의 정신 및 범위 내에 포함될 수 있다. 따라서, 상기 설명 및 도면은 단지 예시적인 것일 뿐이다.

<92> 더욱이, 본 발명의 상기 설명된 실시예는 여러 방식 중의 임의의 방식으로 구현될 수 있다. 예를 들어, 실시예는 하드웨어, 소프트웨어 또는 그 조합을 사용하여 구현될 수 있다. 소프트웨어로 구현될 때, 소프트웨어 코드는 단일 컴퓨터에서 제공되든지 다수의 컴퓨터 사이에서 분산되든지, 임의의 적합한 프로세서 또는 프로세서들의 컬렉션 상에서 실행될 수 있다.

<93> 또한, 여기에서 개략적으로 설명된 다양한 방법 또는 프로세스는 여러 가지 운영 체제 또는 플랫폼 중의 어느 하나를 이용하는 하나 이상의 프로세서상에서 실행가능한 소프트웨어로서 코딩될 수 있다. 부수적으로, 그러한 소프트웨어는 다수의 적합한 프로그래밍 언어 및/또는 종래의 프로그래밍 또는 스크립팅 도구 중의 어느 것을 사용하여 작성될 수 있고, 또한 프레임워크 또는 가상 기계 상에서 실행되는 실행가능 기계 언어 코드 또는 중간 코드로서 컴파일될 수 있다.

<94> 이와 관련하여, 본 발명은 하나 이상의 컴퓨터 또는 기타 프로세서 상에서 실행될 때, 상기 설명된 본 발명의 다양한 실시예를 구현하는 메소드를 실행하는 하나 이상의 프로그램으로 인코드된 하나의 컴퓨터 판독가능 매체(또는 다수의 컴퓨터 판독가능 매체)(예를 들어, 컴퓨터 메모리, 하나 이상의 플로피 디스크, 콤팩트 디스크, 광 디스크, 자기 테이프 등)으로서 구현될 수 있다. 컴퓨터 판독가능 매체 또는 매체들은 저장된 프로그램 또는 프로그램들이 상기 설명된 본 발명의 다양한 실시양상을 구현하기 위해 하나 이상의 상이한 컴퓨터 또는 기타 프로세서상으로 로드될 수 있도록 전송가능하게 될 수 있다.

<95> "프로그램" 또는 "소프트웨어"라는 용어는 상기 설명된 본 발명의 다양한 실시양상을 구현하기 위해 컴퓨터 또는 기타 프로세서를 프로그램하는데 이용될 수 있는 임의 유형의 컴퓨터 코드 또는 컴퓨터 실행가능 명령어 집합을 나타내기 위한 일반적인 의미로 여기에서 사용된다. 부수적으로, 이 실시예의 한 실시양상에 따르면, 실행시에 본 발명의 방법을 실행하는 하나 이상의 컴퓨터 프로그램은 본 발명의 다양한 실시양상을 구현하기

위해, 단일 컴퓨터 또는 프로세서상에 존재해야 하는 것이 아니라, 다수의 상이한 컴퓨터 또는 프로세서 사이에서 모듈러 형태로 분산될 수 있다는 것을 알 수 있을 것이다.

<96> 컴퓨터 실행가능 명령어는 하나 이상의 컴퓨터 또는 기타 장치에 의해 실행된 프로그램 모듈과 같은 다수의 형태로 될 수 있다. 일반적으로, 프로그램 모듈은 특정 태스크를 실행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 개체, 컴포넌트, 데이터 구조 등을 포함한다. 통상적으로, 프로그램 모듈의 기능은 다양한 실시예에서 요구된 대로 결합되거나 분산될 수 있다.

<97> 본 발명의 다양한 실시양상은 단독으로 또는 결합하여 사용될 수 있고, 또는 상기 설명된 실시예에서 구체적으로 설명되지 않은 다양한 배열로 사용될 수 있으므로, 그 애플리케이션에 있어서 상기 명세서에서 설명되거나 도면에 도시된 컴포넌트의 상세 및 배열에 제한되지 않는다. 예를 들어, 한 실시예에서 설명된 실시양상은 다른 실시예에서 설명된 실시양상과 임의의 방식으로 결합될 수 있다.

<98> 청구 요소를 변경하기 위한 청구범위에서의 "제1", "제2", "제3" 등과 같은 일반적인 용어의 사용은 그것만으로 다른 청구 요소에 비교한 한 청구 요소의 임의의 우선순위, 우선권 또는 순서를 내포한다거나, 방법의 액트가 실행되는 시간 순서를 내포하는 것이 아니라, 청구 요소들을 구별하기 위해 소정의 이름을 갖는 한 청구 요소와 동일한 이름을 갖는 다른 청구 요소를 구별하기 위한 표시로서 사용된 것일 뿐이다.

<99> 또한, 여기에서 사용된 어구 및 용어는 설명을 위한 것으로, 제한적인 것으로 간주되어서는 안 된다. 여기에서의 "including", "comprising", 또는 "having", "containing", "involving" 및 그 변형의 사용은 그 이후에 열거된 항목 및 그 등가물뿐만 아니라 추가 항목을 포함하고자 하는 것이다.

도면의 간단한 설명

<10> 첨부 도면은 일정한 비율로 도시된 것은 아니다. 도면에서, 여러 도면에 도시된 각각의 동일한 또는 거의 동일한 컴포넌트는 유사한 참조번호로 표시된다. 명확하게 하기 위해, 모든 컴포넌트가 모든 도면에 표시된 것은 아니다.

<11> 도 1은 종래의 컴퓨터 시스템의 스케치.

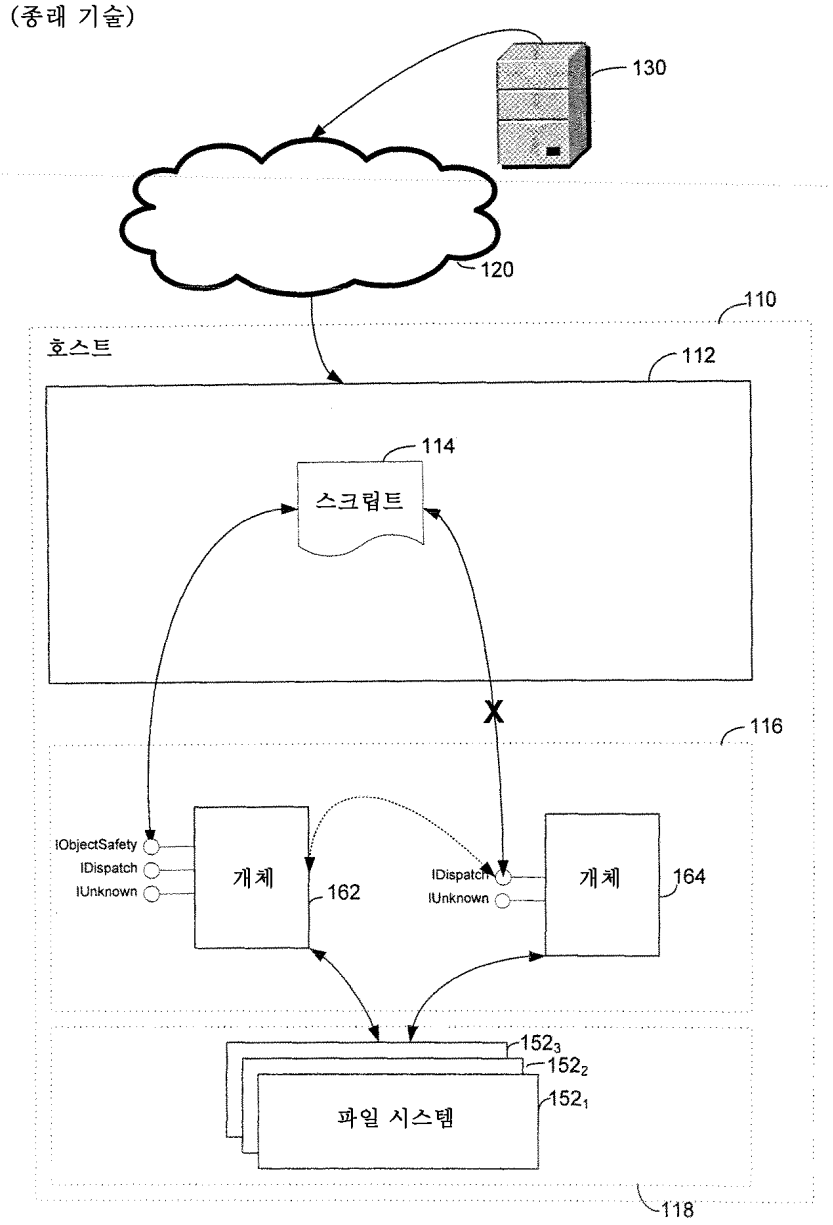
<12> 도 2는 본 발명의 실시예에 따른 컴퓨터 시스템의 동작을 도시한 플로우 차트.

<13> 도 3은 본 발명의 실시예를 포함하는 컴퓨터 시스템의 소프트웨어 블록도.

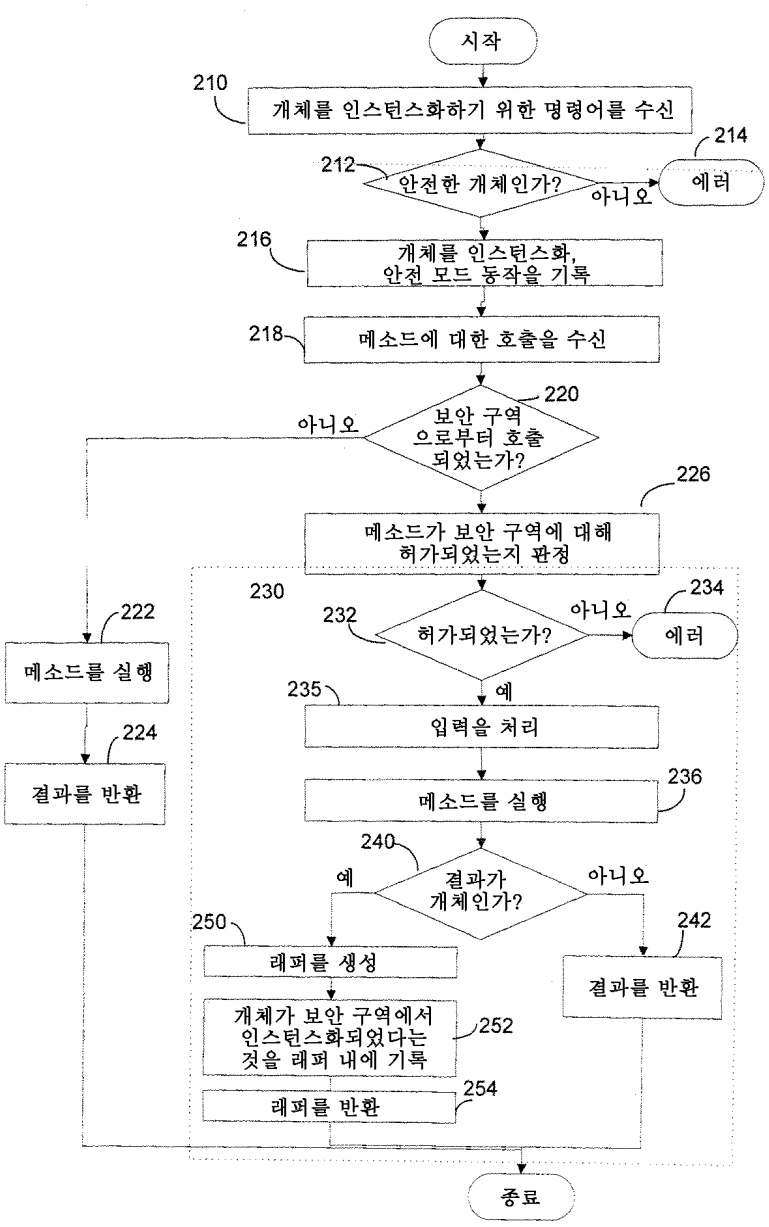
<14> 도 4는 대안적인 동작 상태에서 본 발명의 실시예를 포함하는 컴퓨터 시스템의 소프트웨어 블록도.

도면

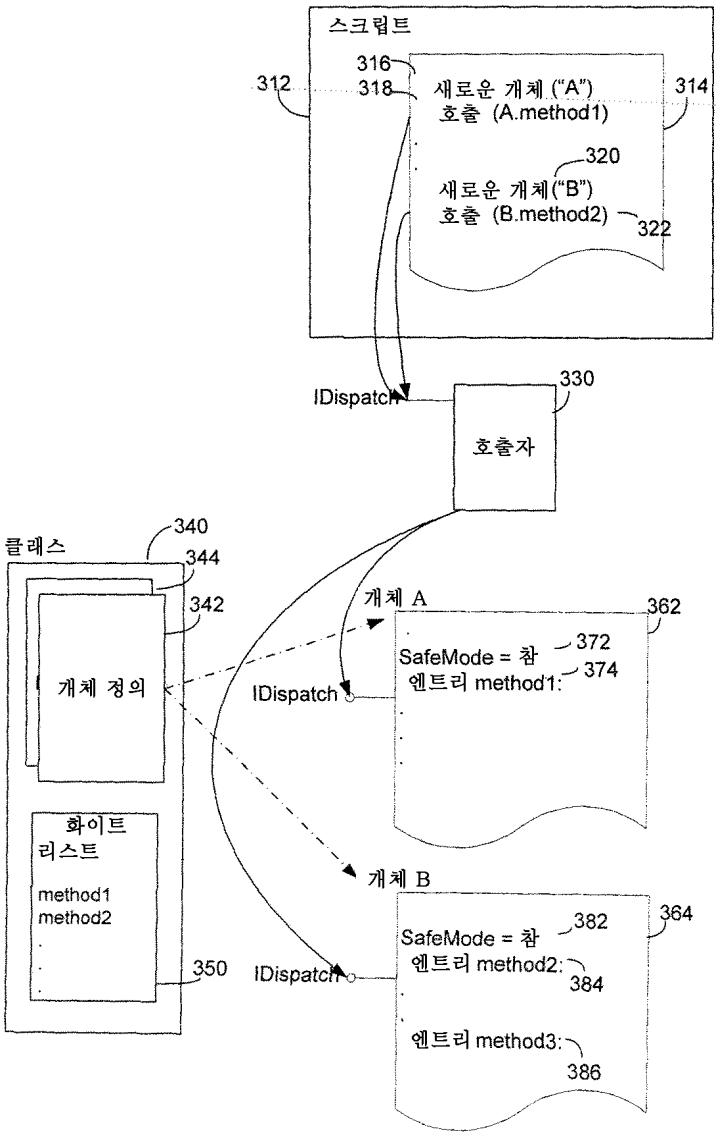
도면1


(종래 기술)
130
120
110
호스트
112
114
스크립트
116
IObjectSafety
IDispatch
IUnknown
개체
162
IDispatch
IUnknown
개체
164
152_3
152_2
152_1
파일 시스템
118

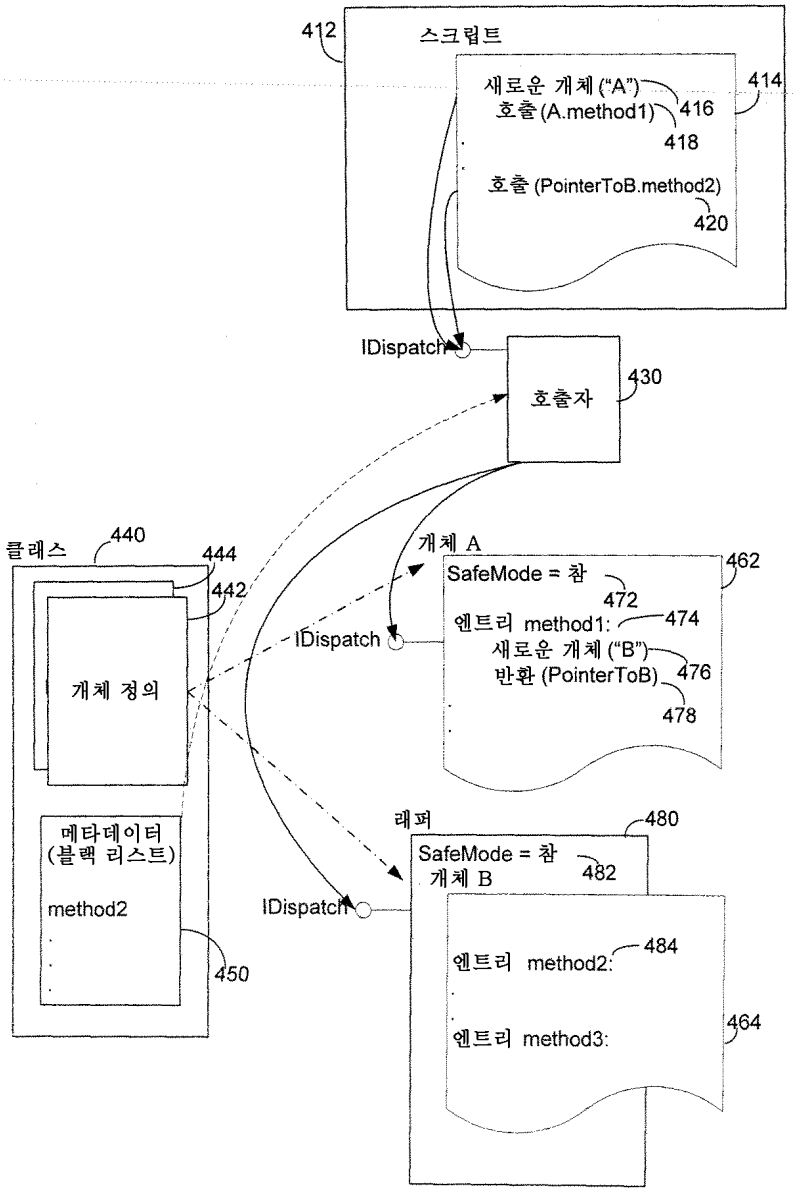
도면2


시작
210
개체를 인스턴스화하기 위한 명령어를 수신
212
안전한 개체인가?
아니오
214
에러
216
개체를 인스턴스화,
안전 모드 동작을 기록
218
메소드에 대한 호출을 수신
220
보안 구역
으로부터 호출
되었는가?
아니오
226
메소드가 보안 구역에 대해
허가되었는지 판정
230
232
허가되었는가?
아니오
234
에러
예
235
입력을 처리
236
메소드를 실행
240
결과가
개체인가?
예
아니오
222
메소드를 실행
224
결과를 반환
250
래퍼를 생성
개체가 보안 구역에서
인스턴스화되었다는
것을 래퍼 내에 기록
252
래퍼를 반환
254
242
결과를 반환
종료

도면3


스크립트
316
312
318
새로운 개체 ("A")
호출 (A.method1)
314
320
새로운 개체("B")
호출 (B.method2)
322
IDispatch
330
호출자
340
클래스
344
342
개체 정의
화이트
리스트
method1
method2
350
IDispatch
개체 A
362
SafeMode = 참
372
엔트리 method1:
374
IDispatch
개체 B
364
SafeMode = 참
382
엔트리 method2:
384
엔트리 method3:
386

도면4


412
스크립트
새로운 개체 ("A")
호출 (A.method1)
416
418
414
호출 (PointerToB.method2)
420
IDispatch
호출자
430
클래스
440
444
442
개체 정의
메타데이터
(블랙 리스트)
method2
450
개체 A
SafeMode = 참
472
462
엔트리 method1:
474
새로운 개체 ("B")
반환 (PointerToB)
476
478
IDispatch
래퍼
480
SafeMode = 참
개체 B
482
엔트리 method2:
484
엔트리 method3:
464
IDispatch