(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0084966
(43) 공개일자 2008년09월22일

(51) Int. Cl.
G06Q 50/00 (2008.03)
(21) 출원번호 10-2008-7014423
(22) 출원일자 2008년06월13일
심사청구일자 없음
번역문제출일자 2008년06월13일
(86) 국제출원번호 PCT/US2006/044691
국제출원일자 2006년11월17일
(87) 국제공개번호 WO 2007/075235
국제공개일자 2007년07월05일
(30) 우선권주장
11/304,300 2005년12월15일 미국(US)

(71) 출원인
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
파샤, 아즈마트, 에이.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
깁슨, 윌리암, 이.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
(74) 대리인
양영준, 백만기

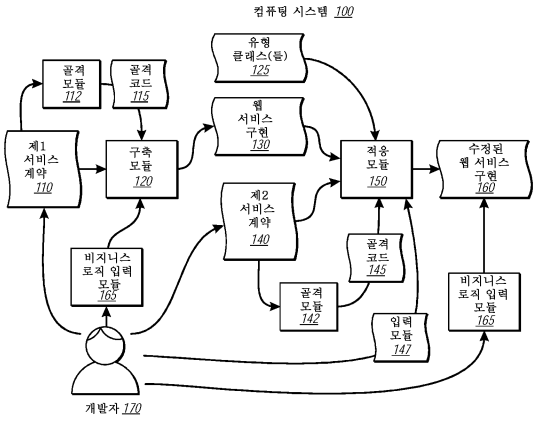
전체 청구항 수 : 총 20 항

(54) 서비스 계약 문서의 웹 서비스 구현 변경 방법 및 이를구현하는 데 사용되는 컴퓨터 프로그램 제품

(57) 요 약

실시예들은 기존 서비스 구현의 부분들을 갱신하기 위한 자동화된 메커니즘을 제공함으로써 서비스 계약의 변경과 연관된 현재의 문제를 처리한다. 예를 들어, 하나의 메커니즘은 NPDL(예를 들어, WSDL) 계약에 대한 변경에 따르도록 구현의 골격을 수정한다. 이러한 실시예에서, 개발자는, 존재할 경우, 통상적으로 계약 정의로부터 알려지지 않는 비지니스 로직에 대한 필요한 변경만을 행하면 된다. 따라서, 이러한 구현에 대한 자동 수정은 개발자의 웹 서비스 개발에 대한 계약 기반 접근법의 채택을 용이하게 한다.

대 표 도 - 도1A



특허청구의 범위

청구항 1

웹 서비스 계약 문서의 서비스 구현-상기 웹 서비스 계약 문서는 상기 서비스 구현이 특정 서비스와 어떻게 통신하는지를 정의함-을 포함하는 컴퓨팅 시스템에서, 상기 웹 서비스 계약 문서의 변경들에 응답하여 상기 서비스 구현의 적어도 일부를 자동으로 변경하는 방법으로서,

웹 서비스 구현을 수신하는 단계 - 상기 웹 서비스 구현은 상기 웹 서비스 구현이 하나 이상의 서비스와 어떻게 통신하는지를 기술하는 제1 웹 서비스 계약 문서에 따라 구축됨 - ;

제2 웹 서비스 계약 문서를 수신하는 단계 - 상기 제2 웹 서비스 계약 문서는 상기 웹 서비스 구현이 상기 제2 계약이 구현될 것을 예상하는 하나 이상의 엔드포인트(endpoint)와 통신할 수 없도록 상기 웹 서비스 구현의 거동에 영향을 미치는 상기 제1 웹 서비스 계약에 대한 하나 이상의 변경을 정의함 - ;

상기 웹 서비스 구현과 상기 제2 웹 서비스 계약 문서 간의 상기 하나 이상의 변경을 식별하는 단계; 및

상기 식별된 하나 이상의 변경에 기초하여, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 적어도 부분적으로 따르도록 자동으로 수정하는 단계

를 포함하는 서비스 구현 변경 방법.

청구항 2

제1항에 있어서, 상기 웹 서비스 구현에 대한 하나 이상의 변경을 식별하는 단계는 상기 웹 서비스 구현의 하나 이상의 부분과 상기 제2 웹 서비스 계약 문서의 하나 이상의 부분을 비교하는 단계를 포함하는 서비스 구현 변경 방법.

청구항 3

제1항에 있어서, 상기 웹 서비스 구현의 적어도 일부의 수정은 상기 웹 서비스 구현에 포함된 하나 이상의 메소드의 하나 이상의 부분을 유지하는(preserve) 서비스 구현 변경 방법.

청구항 4

제3항에 있어서, 상기 웹 서비스 구현에 포함된 하나 이상의 메소드의 하나 이상의 부분은 이들을 자동으로 코멘트 아웃(commenting out)함으로써 유지되는 서비스 구현 변경 방법.

청구항 5

제1항에 있어서,

상기 제2 웹 서비스 계약 문서에 기초하여 골격 코드를 생성하는 단계;

상기 웹 서비스 구현과 상기 골격 코드를 비교하는 단계; 및

상기 비교에 기초하여, 상기 웹 서비스 구현의 적어도 일부를 상기 골격 코드에 따르도록 자동으로 수정하는 단계

를 더 포함하는 서비스 구현 변경 방법.

청구항 6

제1항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하는 단계는 상기 웹 서비스 구현의 결합 명칭 또는 결합 명칭 공간 중 어느 하나가 상기 제2 웹 서비스 계약과 다른 경우에 상기 웹 서비스 구현의 결합 명칭 또는 결합 명칭 공간을 갱신하는 단계를 포함하는 서비스 구현 변경 방법.

청구항 7

제1항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하는 단계는 상기 제2 웹 서비스 계약 문서 내에 포함되지 않은 동작에 대한 웹 메소드를 유지하는 단계를 포

함하는 서비스 구현 변경 방법.

청구항 8

제1항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하는 단계는 상기 제2 웹 서비스 계약에는 포함되어 있지만 상기 웹 서비스 구현에는 포함되지 않은 동작들에 대한 메소드 시그니처 및 속성들을 갖는 메소드를 추가하는 단계를 더 포함하는 서비스 구현 변경 방법.

청구항 9

제1항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하는 단계는 동작에 매칭되는 웹 메소드들에 대한 메소드 시그니처를 수정하는 단계를 포함하는 서비스 구현 변경 방법.

청구항 10

제1항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하는 단계는 메소드에 속성을 추가하여 상기 메소드가 동작으로서 노출되도록 하는 단계를 포함하는 서비스 구현 변경 방법.

청구항 11

제1항에 있어서, 상기 제1 및 제2 웹 서비스 계약 문서들은 웹 서비스 기술 언어(WSDL) 문서들인 서비스 구현 변경 방법.

청구항 12

제1항에 있어서,

상기 제2 웹 서비스 계약 문서에는 존재하지만 상기 제1 웹 서비스 계약 문서에는 존재하지 않는 동작들에 대응하는 상기 웹 서비스 구현에 추가된 웹 메소드들에 대한 적절한 비지니스 로직을 제공하는 단계, 시그니처 또는 메소드 유형이 상기 제2 웹 서비스 계약 문서에 따르도록 수정된 웹 메소드들에 대한 비지니스 로직을 수정하는 단계, 및 동작들이 상기 제1 웹 서비스 계약 문서에는 존재하지만 상기 제2 웹 서비스 계약 문서에는 존재하지 않는 결과로서, 더 이상 상기 동작들로 맵핑되지 않는 웹 메소드들을 제거하는 단계 중 적어도 하나를 개발자가 수행하는 것을 허가하기 위하여, 비지니스 로직 입력 인터페이스를 이용하여 상기 웹 서비스 구현의 수정된 적어도 일부를 노출시키는 단계

를 더 포함하는 서비스 구현 변경 방법.

청구항 13

웹 서비스 계약 문서의 웹 서비스 구현-상기 웹 서비스 계약 문서는 상기 서비스 구현이 특정 서비스와 어떻게 통신하는지를 정의함-을 포함하는 컴퓨팅 시스템에서, 상기 웹 서비스 계약 문서의 변경들에 응답하여 상기 웹 서비스 구현의 적어도 일부를 자동으로 변경하는 방법을 구현하는 데 사용되는 컴퓨터 프로그램 제품으로서,

컴퓨터 실행가능 명령어들을 저장한 하나 이상의 컴퓨터 판독가능 매체

를 포함하고,

상기 컴퓨터 실행가능 명령어들은 상기 컴퓨팅 시스템의 하나 이상의 프로세서에 의해 실행될 때 상기 컴퓨팅 시스템으로 하여금

웹 서비스 구현을 수신하는 것 - 상기 웹 서비스 구현은 상기 웹 서비스 구현이 하나 이상의 서비스와 어떻게 통신하는지를 기술하는 제1 웹 서비스 계약 문서에 따라 구축됨 - ;

제2 웹 서비스 계약 문서를 수신하는 것 - 상기 제2 웹 서비스 계약 문서는 상기 웹 서비스 구현이 상기 제2 계약이 구현될 것을 예상하는 하나 이상의 엔드포인트와 통신할 수 없도록 상기 웹 서비스 구현의 거동에 영향을 미치는 상기 제1 웹 서비스 계약에 대한 하나 이상의 변경을 정의함 - ;

상기 웹 서비스 구현과 상기 제2 웹 서비스 계약 문서 간의 상기 하나 이상의 변경을 식별하는 것; 및

상기 식별된 하나 이상의 변경에 기초하여, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 적어도 부분적으로 따르도록 자동으로 수정하는 것

을 수행하게 하는 컴퓨터 프로그램 제품.

청구항 14

제13항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하도록 구성되는 컴퓨터 실행가능 명령어는 결합 명칭 또는 결합 명칭 공간이 상기 제2 웹 서비스 계약과 다른 경우에 웹 서비스의 결합 속성을 갱신하도록 구성되는 명령어를 더 포함하는 컴퓨터 프로그램 제품.

청구항 15

제13항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하도록 구성되는 컴퓨터 실행가능 명령어는 상기 제2 웹 서비스 계약 문서에 포함되지 않은 동작에 대한 웹 메소드 속성을 코멘트 아웃하도록 구성되는 명령어를 더 포함하는 컴퓨터 프로그램 제품.

청구항 16

제13항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하도록 구성되는 컴퓨터 실행가능 명령어는 상기 제2 웹 서비스 계약에는 포함되어 있지만 상기 웹 서비스 구현에는 포함되지 않은 동작들에 대한 메소드 시그니처 및 속성들을 갖는 웹 메소드를 추가하도록 구성되는 명령어를 더 포함하는 컴퓨터 프로그램 제품.

청구항 17

웹 서비스 계약 문서의 웹 서비스 구현 - 상기 웹 서비스 계약 문서는 상기 웹 서비스 구현이 특정 서비스와 어떻게 통신하는지를 정의함 - 을 포함하는 컴퓨팅 시스템에서, 상기 웹 서비스 계약 문서의 변경들에 응답하여 상기 서비스 구현의 적어도 일부를 자동으로 변경하는 방법으로서,

상기 웹 서비스 구현이 하나 이상의 서비스와 어떻게 통신하는지를 기술하는 제1 웹 서비스 계약 문서에 대한 변경들을 식별함으로써 웹 서비스 구현의 적어도 일부를 제2 웹 서비스 계약에 적어도 부분적으로 따르도록 자동으로 수정하는 단계

를 포함하고,

상기 제2 웹 서비스 계약에서 이루어진 변경들은 상기 웹 서비스 구현이 상기 제2 계약이 구현될 것으로 예상하는 하나 이상의 엔드포인트와 통신할 수 없도록 상기 웹 서비스 구현의 거동에 영향을 미치는 서비스 구현 변경 방법.

청구항 18

제17항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하는 단계는 동작에 매칭되는 웹 메소드들에 대한 메소드 시그니처를 수정하는 단계를 더 포함하는 서비스 구현 변경 방법.

청구항 19

제17항에 있어서, 상기 웹 서비스 구현의 적어도 일부를 상기 제2 웹 서비스 계약 문서에 따르도록 자동으로 수정하는 단계는 메소드가 상기 웹 서비스 구현으로부터 제거되지는 않지만, 상기 메소드가 동작으로서 노출되지도 않도록 상기 메소드에 대한 속성을 제거하는 단계를 포함하는 서비스 구현 변경 방법.

청구항 20

제17항에 있어서,

상기 제2 웹 서비스 계약 문서에는 존재하지만 상기 제1 웹 서비스 계약 문서에는 존재하지 않는 동작들에 대응하는 상기 웹 서비스 구현에 추가된 웹 메소드들에 대한 적절한 비지니스 로직을 제공하는 단계, 시그니처 또는 메소드 유형이 상기 제2 웹 서비스 계약 문서에 따르도록 수정된 웹 메소드들에 대한 비지니스 로직을 수정하는 단계, 및 동작들이 상기 제1 웹 서비스 계약 문서에는 존재하지만 상기 제2 웹 서비스 계약 문서에는 존재하지

않는 결과로서, 더 이상 상기 동작들로 맵핑되지 않는 웹 메소드들을 제거하는 단계 중 적어도 하나를 개발자가 수행하는 것을 허가하기 위하여, 비지니스 로직 입력 인터페이스를 이용하여 상기 웹 서비스 구현의 수정된 적어도 일부를 노출시키는 단계

를 더 포함하는 서비스 구현 변경 방법.

명 세 서

기 술 분 야

<1> 본 발명은 웹 서비스 계약 문서의 변경에 따라 서비스 구현을 자동으로 수정 또는 갱신하기 위한 방법, 시스템 및 컴퓨터 프로그램 제품에 관한 것이다.

배 경 기 술

<2> 컴퓨터 시스템 및 관련 기술은 많은 사회 양상에 영향을 미친다. 실로, 컴퓨터 시스템의 정보 처리 능력은 우리가 생활하고 일하는 방식을 바꾸었다. 현재, 컴퓨터 시스템은 일반적으로, 컴퓨터 시스템의 출현 이전에는 수동으로 수행되었던 많은 작업(예를 들어, 워드 프로세싱, 스케쥴링, 데이터베이스 관리 등)을 수행한다. 최근에, 컴퓨터 시스템들은 이들이 데이터를 공유하기 위해 전자적으로 통신할 수 있는 컴퓨터 네트워크를 형성하기 위해 서로 결합되었다. 웹 서비스들은 이러한 컴퓨터 시스템들 간의 통신을 촉진시키는 동력이었다.

<3> 웹 서비스들은 애플리케이션들이 데이터를 공유하고, 보다 강력하게는 다른 애플리케이션들이 어떻게 구축되어 있는지, 이들이 어떤 운영 체제 또는 플랫폼 상에서 실행되는지, 그리고 이들에 액세스하기 위해 어떤 디바이스들이 사용되는지에 관계없이 이들로부터 능력들을 호출하게 한다. 웹 서비스들은 SOAP(Simple Open Access Protocol), XML(eXtensible Markup Language), UDDI(Universal Description Discovery Integration), WSDL(Web Service Description Language) 등을 포함하는 산업 표준 프로토콜들에 의해 인터넷을 통해 호출될 수 있다. 웹 서비스들은 서로 독립적으로 유지되지만, 이들은 그 자신들을 특정 작업을 수행하는 협력 그룹으로 느슨하게 링크할 수 있다.

<4> 종종, 웹 서비스 네트워크 상에서의 전자 통신은 서버 컴퓨터 시스템(이하, "서버", "서비스" 또는 "웹 서비스"라고 함)에서의 네트워크 서비스(예를 들어, 웹 서비스)에 대한 액세스를 요청하는 클라이언트 컴퓨터 시스템(이하, "클라이언트"라 함)을 포함한다. 따라서, 클라이언트는 서비스의 시스템 자원들에 대한 특정 액세스를 위해 서비스에 요청을 전송하며, 서비스는 원하는 정보를 제공하는 응답 메시지로 응답한다. 물론, 클라이언트와 서비스 간의 다른 메시징 패턴들 또한 이용될 수 있으며, 간단한 단독 메시지들은 물론, 예를 들어 통지, 청구-응답, 퍼브-서브 패턴, 폴링, 킥-푸시, 큐잉 등과 같은 보다 복잡한 멀티 메시지 교환을 포함할 수 있다. 또한, 인증 및 검증 메커니즘들과 같이, 서비스 자원들에 액세스하는 데 필요한 다른 프로시저들이 존재할 수 있다.

<5> 서비스에 액세스하는 데 사용되는 메시지 패턴들 및/또는 프로시저들의 유형에 관계없이, 웹 서비스들은 플랫폼 불가지론 방식으로 연동성을 제공하는데, 이는 클라이언트 및 서비스 양자가 기본 계약에 동의하고 있기 때문이다. 기계 판독가능 언어와 인간 언어의 혼합에 의해 표현되는 이러한 계약들은 특히 웹 서비스에 의해 제공되는 동작들-기능의 단위들- 및/또는 호출시 각각의 동작으로 전달되고 그리고/또는 각각의 동작에 의해 반환되는 메시지들의 포맷을 정의한다. 네트워크 프로토콜 기술 언어(예를 들어, WSDL)는 공통 또는 표준 언어로 웹 서비스 계약을 기술하기 위한 전반적인 랩퍼 또는 사양을 제공한다. 이러한 사양은 개발자 및 개발자 도구가 계약을 생성하고 해석하는 것을 용이하게 하며, 또한 많은 부분에서 그들의 대중성을 설명하는 포괄적인 도구 모음을 갖는다.

<6> 웹 서비스에 대한 계약은 서비스 구현과 무관하게 작성되거나(예를 들어, 다른 조직으로부터의 비지니스 로직을 포함하는 계약), 소정의 자동화된 수단에 의해 기존 서비스의 거동을 반영하도록 생성될 수 있다. 점차, 개발자들은 구현과 무관하게 계약을 정확히 진술할 수 있는 능력을 필요로 한다. 예를 들어, 다수의 기업, 또는 동일 기업 내의 그룹들 또는 팀들 간의 연동성 시나리오는 종종 "계약 선행" 또는 "계약 기반" 개발 전략을 이용한다. 이러한 사례에서, 당사자들 간에 계약이 합의된 경우, 다양한 엔드포인트(endpoint)의 비지니스 로직에 따르도록 웹 서비스의 특정 구현(들)이 생성될 수 있다. 기존의 도구들은 웹 서비스 계약에 기초하여 "골격" 웹 서비스 구현의 생성을 가능하게 하며, 이어서 이것은 개발자가 서비스에 의해 제공되는 각각의 동작에 대한 비지니스 로직의 상세를 완성하는 것을 가능하게 한다. 통상적으로, 이러한 골격 구현은 계약에 의해 정의되는 바와 같은 각각의 동작에 대한 메소드들을 갖는 서비스에 대한 클래스를 포함한다. 또한, 각각의 메소드의 시

그니처는 동작으로 전송되고 그리고/또는 동작에 의해 반환되는 메시지들을 기술하며, 시그니처에서 참조되는 메시지 데이터 구조들을 기술하는 데 필요할 경우에는 추가 클래스들이 생성된다.

<7> 그러나, 종종, 웹 서비스 계약은 웹 서비스 구현이 시작된 후에 변경된다. 예를 들어, 새로운 또는 수정된 비지니스 요건을 만족시키기 위하여, 또는 개발자들이 반복 또는 증식 개발 전략을 이용하는 경우에 계약은 변경을 필요로 할 수 있다. 따라서, 웹 서비스 계약의 변경은 통상적으로, 갱신된 서비스 계약에 따르기 위해 골격(메소드, 시그니처, 및 클래스 또는 데이터 구조) 및 비지니스 로직 양자에 대한 변경을 필요로 한다. 그러나, 현재의 도구들은 계약의 변경을 수용하기 위한 웹 서비스 구현의 갱신을 지원하지 않는다.

<8> 따라서, 개발자는 갱신된 웹 서비스 계약에 따르도록 웹 서비스 구현을 수정하는 데 이용할 수 있는 옵션을 거의 갖지 못한다. 예를 들어, 개발자는 통상적으로 (1) 기존의 도구들(예를 들어, 골격 도구)을 이용하여 개정된 계약으로부터 새로운 서비스 구현을 재생성하고, 유지하기를 원하는 이전 서비스 구현의 양태들을 수동으로 이동시키거나, (2) 기존 구현 코드를 수동으로 편집하여, 개정된 계약에 따르는 데 필요한 변경을 행할 것이다. 변경의 범위에 따라, 양 접근법은 노동 집약적이고 인간적인 에러가 발생하기 쉽다. 또한, 이러한 접근법들은 개발자가 사용되는 NPDL 언어(예를 들어, WSDL) 및/또는 스키마(예를 들어, XML 스키마)에 대한 깊은 지식을 가져서 변경의 특성 및 이것이 구현에 어떻게 맵핑되는지를 이해하는 것을 필요로 할 수 있다. 그러나, 이러한 언어들 및 스키마들의 복잡성으로 인해, 이러한 수정은 대부분의 개발자들의 통상적인 기술의 범위를 넘는다.

발명의 상세한 설명

<13> 계약 변경에 응답하여 서비스 구현을 수정하기 위한 현재 메커니즘들의 전술한 결함들 및 결점들은 여기에 설명되는 실시예들을 통해 극복된다. 본 요약은 아래의 상세한 설명에서 더 설명되는 개념들의 선택을 간략한 형태로 소개하기 위해 제공된다는 점에 유의한다. 본 요약은 청구 발명의 주요 특징들 또는 본질적 특징들을 식별하고자 하는 의도도 없고, 청구 발명의 범위를 결정하는 데 있어서 보조물로 사용하고자 하는 의도도 없다.

<14> 하나의 실시예는 웹 서비스 계약 문서의 변경에 응답하여 서비스 구현의 적어도 일부를 자동으로 변경하는 것을 제공한다. 이 실시예에서, 제1 웹 서비스 계약 문서에 따라 구성된 웹 서비스 구현이 수신된다. 제1 웹 서비스 계약 문서는 웹 서비스 구현이 어떻게 서비스(들)와 통신할지를 기술한다. 또한, 제1 웹 서비스 계약 문서와 다른 제2 웹 서비스 계약 문서가 또한 수신된다. 하나 이상의 변경은 웹 서비스 구현이 제2 계약이 구현될 것으로 예측하는 엔드포인트들과 통신할 수 없는 방식으로 웹 서비스 구현의 거동에 영향을 미친다. 이후, 웹 서비스 구현과 제2 계약 문서 간의 하나 이상의 변경이 식별된다. 이러한 식별에 기초하여, 웹 서비스 구현이 제2 웹 서비스 계약 문서에 적어도 부분적으로 따르도록 웹 서비스 구현의 일부가 자동으로 수정된다.

<15> 여기에 설명되는 실시예들의 추가적인 특징들 및 이점들은 이어지는 설명에서 설명되며, 부분적으로는 설명으로부터 자명하거나 발명의 실시에 의해 알 수 있다. 여기에 설명되는 실시예들의 특징들 및 이점들은 첨부된 청구항들에서 구체적으로 지시되는 기구들 및 조합들에 의해 실현되고 얻어질 수 있다. 여기에 설명되는 실시예들의 이들 및 다른 특징들은 아래의 설명 및 첨부된 청구항들로부터 더 충분히 명확해지거나, 이후에 설명되는 바와 같은, 여기에 설명되는 실시예들의 실시에 의해 알 수 있다.

실 시 예

<16> 본 발명의 상기 및 다른 이점들 및 특징들이 얻어질 수 있는 방법을 설명하기 위하여, 위에 간략하게 설명된 본 발명의 보다 구체적인 설명이 첨부된 도면들에 도시된 본 발명의 특정 실시예들을 참조하여 행해질 것이다. 이 도면들은 본 발명의 대표적인 실시예들만을 도시하고 있고, 따라서 본 발명의 범위를 제한하는 것으로 간주되지 않아야 한다는 것을 이해하면서, 본 발명을 첨부 도면들을 이용하여 더 구체적이고 상세하게 기술하고 설명할 것이다.

<17> 본 발명은 웹 서비스 계약 문서의 변경에 따라 서비스 구현을 자동으로 수정 또는 갱신하기 위한 방법, 시스템 및 컴퓨터 프로그램 제품에 관한 것이다. 본 발명의 실시예들은 후술하는 바와 같이 다양한 컴퓨터 하드웨어 또는 모듈을 포함하는 특수 목적 또는 범용 컴퓨터를 포함할 수 있다.

<18> 실시예들은 기존 서비스 구현의 부분들을 갱신하기 위한 자동화된 메커니즘을 제공함으로써 서비스 계약의 변경과 연관된 현재의 문제를 처리한다. 예를 들어, 하나의 메커니즘은 NPDL(예를 들어, WSDL) 계약에 대한 변경에 따르도록 구현의 골격을 수정한다. 이러한 실시예에서, 개발자는, 존재할 경우, 통상적으로 계약 정의로부터 알려지지 않는 비지니스 로직에 대한 필요한 변경만을 행하면 된다. 따라서, 이러한 구현에 대한 자동 수정은 개발자의 웹 서비스 개발에 대한 계약 기반 접근법의 채택을 용이하게 한다.

<19> 예를 들어, 일 실시예는 서비스 구현; 웹 서비스에 의해 사용되는 클래스(들)의 유형; 웹 서비스 계약; 및/또는 다른 입력 정보(예를 들어, 결합을 위한 맵핑 정보)를 입력으로 취한다. 웹 서비스에 의해 현재 구현되고 있는 계약과 다른 지정된 웹 서비스 계약을 이용하여 현재 구현이 갱신되어, 그의 클래스, 클래스 속성, 메소드, 메소드 속성, 메소드 시그니처 및 다른 다양한 데이터 구조 또는 특성이 지정된 계약을 따르게 된다. 그러나, 갱신된 웹 서비스 구현은 계약에 따른 후에 컴파일 또는 실행되지 않을 수도 있다는 점에 유의한다. 따라서, 프로세스를 완료하기 위하여, 개발자는 이루어진 변경들을 검사할 필요가 있으며, (1) 새로운 계약에는 있지만 이전 계약에는 없는 동작들에 대응하는 추가 메소드들에 대한 적절한 비지니스 로직 및/또는 메소드 본문 코드를 제공하는 것, (2) 시그니처가 변경되었고 새로운 또는 변경된 거동을 필요로 하며 그리고/또는 시그니처 내의 새로운 또는 변경된 유형들을 참조하는 메소드들에 대해 요구되는 바와 같이 비지니스 로직 및/또는 메소드 본문 코드를 수정하는 것, 또는 (3) 이전에 존재하는 동작들이 계약으로부터 제거된 결과로서 더 이상 동작들로 맵핑되지 않는 메소드들을 선택적으로 제거하는 것 중 하나 이상을 행할 수 있다.

<20> 여기에 설명되는 실시예들은 귀중한 정보를 포함하거나 사용자가 소정의 다른 상황에서 사용하기를 원할 수 있는 상당한 노력의 결과를 나타낼 수 있는 갱신된 구현 내의 메소드들의 부분들(예를 들어, 사용자가 작성한 비지니스 로직)의 유지(preservation)를 가능하게 한다는 점에 유의한다. 예를 들어, 구현의 변경이 동작의 재명명으로부터 발생하는 경우, 유지되는 메소드 본문 코드가 새로 명명된 메소드로 이동되는 것만이 필요할 수 있다. 또한, 메소드의 유지는 메소드(들), 메소드 속성(들), 메소드 시그니처 등에 대한 소정 형태로 코멘트 아웃(some form of commenting out)한 것일 수 있다. 또한, 다른 실시예들은 구현을 갱신된 계약에 따르게 하는 프로세스 동안 코멘트의 생성을 가능하게 하여, 개발자가 전반적인 서비스 갱신 활동을 완료하는 데 필요한 작업을 식별하는 것을 훨씬 더 쉽게 해준다.

<21> 이로운 특징들에 대한 보다 구체적인 참조가 도면들과 관련하여 후술되지만, 본 발명의 범위 내의 실시예들은 또한 컴퓨터 실행가능 명령어 또는 데이터 구조를 지니거나 저장하기 위한 컴퓨터 판독가능 매체를 포함한다. 이러한 컴퓨터 판독가능 매체는 범용 또는 특수 목적 컴퓨터에 의해 액세스될 수 있는 임의의 이용 가능 매체일 수 있다. 예를 들어, 이러한 컴퓨터 판독가능 매체는 RAM, ROM, EEPROM, CD-ROM 또는 다른 광 디스크 저장 장치, 자기 디스크 저장 또는 다른 자기 저장 장치, 또는 컴퓨터 실행가능 명령 또는 데이터 구조의 형태로 원하는 프로그램 코드 수단을 지니거나 저장하는 데 사용될 수 있고 범용 또는 특수 목적 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있지만, 이에 제한되는 것은 아니다. 정보가 네트워크 또는 다른 통신 접속(유선, 무선 또는 유선 및 무선의 조합)을 통해 컴퓨터에 전송 또는 제공될 때, 컴퓨터는 접속을 컴퓨터 판독가능 매체로서 적절히 간주한다. 따라서, 이러한 임의의 접속은 적절히 컴퓨터 판독가능 매체라고 한다. 위의 조합들 또한 컴퓨터 판독가능 매체의 범위 내에 포함되어야 한다.

<22> 컴퓨터 실행가능 명령은 예를 들어 범용 컴퓨터, 특수 목적 컴퓨터, 또는 특수 목적 처리 장치가 소정의 기능 또는 기능들의 그룹을 수행하게 하는 명령 및 데이터를 포함한다. 본 발명은 구조적 특징들 및/또는 방법론적 동작들에 고유한 언어로 설명되지만, 첨부된 청구범위에서 정의되는 발명은 전술한 특정 특징들 또는 동작들로 제한될 필요는 없다. 오히려, 전술한 특정 특징들 및 동작들은 청구범위를 구현하는 예시적인 형태들로서 개시된다.

<23> 본 명세서에서 사용될 때, "모듈" 또는 "컴포넌트"라는 용어는 컴퓨팅 시스템 상에서 실행되는 소프트웨어 개체들 또는 루틴들을 지칭할 수 있다. 여기에 설명되는 상이한 컴포넌트들, 모듈들, 엔진들 및 서비스들은 컴퓨팅 시스템 상에서 실행되는 개체들 또는 프로세스들로서(예를 들어, 개별 스레드들로서) 구현될 수 있다. 여기에 설명되는 시스템 및 방법들은 소프트웨어로 구현되는 것이 바람직하지만, 하드웨어 또는 소프트웨어와 하드웨어의 조합에 의한 구현도 가능하며 고려된다. 본 설명에서, "컴퓨팅 엔티티"는 본 명세서에서 앞서 정의된 바와 같은 임의의 컴퓨팅 시스템, 또는 컴퓨팅 시스템 상에서 실행되는 임의의 모듈 또는 모듈들의 조합일 수 있다.

<24> 도 1A는 갱신된 웹 서비스 계약 문서에 따르도록 웹 서비스 구현의 적어도 일부를 자동으로 수정하기 위한 컴퓨팅 시스템(100)을 나타낸다. 컴퓨팅 시스템(100)은 실시예를 구현할 때 사용될 수 있는 다양한 모듈 및 컴포넌트를 도시하고 있다. 이러한 다양한 모듈 및 컴포넌트는 단일 모듈로 결합되고, 소프트웨어, 하드웨어 또는 이들의 조합을 이용하여 구현될 수 있다. 또한, 이 모듈들/컴포넌트들은 동일 컴퓨터 내에 포함되거나 임의 수의 시스템들에 분산될 수 있다. 물론, 여기에 설명되는 실시예들에 의해 제공되는 기능을 수행하기 위한 다른 구성들 및 메커니즘들도 실시예들을 제공하는 데 이용될 수 있다. 따라서, 컴퓨팅 시스템(100)은 예시적인 목적으로만 사용되며, 여기에 설명되는 실시예들의 범위를 제한하거나 축소하는 것을 의도하지 않는다.

<25> 컴퓨터 시스템(100)의 유형 또는 구성과 관계없이, 컴퓨팅 시스템(100)은 웹 서비스 구현이 그 안에 제공되는

서비스들 또는 엔드포인트들과 어떻게 통신할지를 정의하는 제1 계약 또는 기술(description)(여기서 교환 가능하게 사용됨) 문서(110)를 수신 또는 액세스한다(여기서 사용되는 "엔드포인트"라는 용어는 계약에 따라 서비스와 통신하도록 구성된 클라이언트, 서비스 또는 임의의 다른 엔티티를 포함할 수 있다는 점에 유의한다). 전술한 바와 같이, 몇몇 실시예에서, 제1 계약 문서(110)는 웹 서비스 구현(130)의 구축 이전에 하나 이상의 개발자(170)에 의해 합의 및 생성될 수 있다. 개발자(170)는 개별 개발자(170) 또는 둘 이상의 개발자(170) 또는 기업의 팀인 것으로 간주될 수 있으며, 정확한 수 또는 구성은 여기에 설명되는 실시예들에 중요하지 않다. 또한, 전술한 엔티티들 중 어느 하나가(개별적으로 또는 조합하여) 개발자(170)에 의해 생성되는 것으로 설명되는 다양한 문서를 생성할 수 있다.

<26> 누가 또는 어떻게 웹 서비스 기술(110)을 생성하는지에 관계없이, 통상적으로 제1 계약(110)은 제공되는 하나 이상의 서비스의 거동을 기술할 것이다. 따라서, 제1 기술(110)은 특히, 제공되는 동작(들), 교환되는 메시지의 포맷은 물론, 다른 특성들을 기술한다. 이 리스트는 포괄적도 총괄적도 아니라는 점에 유의한다. 예를 들어, 서비스의 동작은 입력 메시지를 수신하지만, 출력 메시지를 반환하지 않을 수 있다. 또한, 여기에 리스트되지 않은 제1 (및 다른) 기술(110) 내에는 다른 거동들 및 데이터 개체들이 존재할 수 있다. 따라서, 여기에 제공되는 거동들 및 데이터 개체들의 리스트는 단지 예시적이며, 명시적으로 청구되지 않는 한은 실시예들의 범위를 제한 또는 축소하는 것을 의도하지 않는다.

<27> 또한, 도 1의 웹 서비스 계약 문서들(예를 들어, 제1 계약 문서(110) 및 제2 계약 문서(140))은 WSDL 계약 문서들로서 지정될 수 있는데, 이는 이러한 포맷이 이 분야에서 널리 사용되기 때문이라는 점에 유의한다. 그러나, 전술한 바와 같이, WSDL은 NPDL을 위한 하나의 예시적인 포맷일 뿐이다. 따라서, WSDL 지정은 명시적으로 청구되지 않는 한은 개시되는 실시예들 또는 첨부된 청구항들의 범위를 제한하는 데 사용되어서는 안 된다.

<28> 그럼에도, 웹 서비스 계약(110)은 웹 서비스 구현(130)을 생성하는 데 사용될 수 있다. 예를 들어, 구축 모듈(120)은 제1 웹 서비스 기술(110) 내에 포함된 데이터를 취하여 특정 구현(130)을 생성하는 데 사용될 수 있다. 도 1B에 도시된 바와 같이, 웹 서비스 구현(130)은 서비스에 의해 사용되는 웹 서비스 클래스 유형(들)(132)을 포함할 수 있는데, 이는 통상적으로 그와 관련된 다양한 클래스 속성(131)을 갖는다. 웹 서비스 클래스(132)는 제공되는 다양한 메소드(133)를 포함하는 웹 서비스의 실제 구현(130)을 정의한다. 전술한 바와 같이, 웹 서비스들은 이들이 서비스 계약(예를 들어, 제1 계약 문서(110)) 내에 정의된 바와 같은 동작들을 이용하여 수용하는 메시지들의 유형을 정의한다. 이러한 동작들은 웹 서비스 구현(130) 내의 웹 서비스 메소드(들)(133) 각각에 상관된다.

<29> 각각의 메소드(133)는 그와 연관된 다양한 메소드 속성(134) 및 다른 데이터 구조들 및 특성들(137)을 갖는다. 또한, 메소드(133)는 통상적으로 메소드(133)의 메소드 명칭, 메소드 유형, 예측 입력 파라미터(들) 및/또는 반환 유형(들)과 같은 것들을 정의하는 메소드 시그니처(135)를 더 포함한다. 또한, 후술하는 바와 같이, 메소드 시그니처는 통상적으로 개발자(170)에 의해 정의되는 바와 같은 비지니스 로직(136)을 포함할 것이다. 물론, 메시징 포맷(예를 들어, SOAP), 결합 프로토콜 등과 같은 다른 특성들도 메소드(133), 메소드 속성들(134), 메소드 시그니처(135) 및/또는 다른 데이터 구조들 또는 특성들(137) 내에 정의될 수 있다.

<30> 예를 들어, 구축 모듈(120)을 이용하여 생성되는 웹 서비스 구현(130)은 다양한 클래스 속성(131) 및 메소드(133)를 갖는 클래스 유형(132) "산술"일 수 있지만 이에 제한되는 것은 아니다. 예를 들어, 이 사례에서 메소드들(133)은 "가산", "감산", "승산" "제산" 등을 포함할 수 있다. 메소드 속성들(134), 메소드 시그니처(135) 및 다른 데이터 구조들 또는 특성들(137)은 각각의 메소드(133)에 대해 특정 동작의 명칭(예를 들어, 나누기), 메소드 유형(예를 들어, 제산), 입력에 필요한 정수들의 수(예를 들어, 2개), 결과되는 출력의 유형(예를 들어, 단일 INT), 입력 및/또는 출력 양자의 포맷(예를 들어, SOAP 메시지)은 물론, 임의의 다른 적절한 특성들을 더 정의할 수 있다.

<31> 서비스를 구현하는 클래스 유형(132)의 메소드(들)(133)는 대개는 웹 서비스 정의(예를 들어, 제1 계약 문서(110)) 내의 동작으로 자동 맵핑되지 않는다는 점에 유의한다. 예를 들어, 통상적인 도구들(예를 들어, 골격 모듈(112))은 완전한 구현에 필요한 비지니스 로직(136) 없이 메소드(들)(133), 메소드 속성(들)(134), 메소드 시그니처들(135), 클래스 유형들(132) 등을 특히 정의하는 골격 코드(115)를 생성한다. 결과적으로, 개발자(170)는 통상적으로 서비스 정의에서 서비스 요청들 및 응답들로서 어느 메소드들(132)이 노출되어야 하는지를 (예를 들어, 비지니스 로직 입력 모듈(165)을 이용하여) 수동으로 기술하는 것이 필요할 것이다. 또한, 개발자(170)는 서비스를 위해 제1 계약(110)에 의해 정의되는 바와 같은 각각의 동작에 대한 비지니스 로직(136)의 상세를 완성하는 것이 필요하다. 그러나, 몇몇 실시예에서, 골격 코드(115)를 생성한 후 웹 서비스 구현(130)을

생성하는 중간 단계는 옵션(option)이라는 점에 유의한다.

<32> 전술한 바와 같이, 구축 모듈(120)은 웹 서비스 구현(130)을 생성한다. 웹 서비스 구현(130)은 제1 NPDL(예를 들어, WSDL)에 의해 지시되는 바와 같은 클래스들, 메소드들, 메소드 속성들, 웹 시그니처들, 메시지 유형들 등을 포함할 수 있으며, 제공하려고 하는 웹 서비스의 거동들을 구현하는 특정 비지니스 코드를 또한 포함할 수 있다. 컴파일시, 웹 서비스 구현(130)은 NPDL 계약에 의해 지시된 거동들을 수행해야 한다.

<33> 전술한 바와 같이, 어떻게 웹 서비스 구현(130)이 생성되는지에 관계없이, 제1 서비스 계약(110)은 웹 서비스 구현(130)이 생성된 후에 변경될 수 있다. 예를 들어, 개발자(170)는 제1 서비스 계약(110)을 갱신, 수정 또는 완전히 대체하는 제2 서비스 계약(140)을 생성할 수 있다. 이러한 변경들은 특히, 제공되는 동작(들)의 변경, 교환되는 메시지들의 포맷은 물론, 다른 특성들을 포함할 수 있다. 따라서, 제1 계약(110)을 이용하여 생성된 웹 서비스 구현(130)을 이용하는 엔드포인트들은 제2 계약(140) 내에 지정된 서비스들과 더 이상 통신하지 못할 수 있다. (그러나, 몇몇 사례에서, 제1 계약(110)에 대한 변경은 최초 서비스 구현(130)을 이용하는 엔드포인트가 제2 계약(140) 내에 제공된 서비스들과 통신할 수 없다는 것을 반드시 의미하는 것은 아니라는 점에 유의한다.)

<34> 예를 들어, 위의 "산술" 클래스 유형의 예에서, 개발자는 입력들의 수 및/또는 유형을 후속 변경할 수 있다. 예를 들어, 제1 계약(110)은 "가산"의 메소드가 입력들의 오픈 엔디드 어레이(open ended array)를 갖는 것을 허가하는 것으로 가정한다. 개발자(170)는 서비스 공격의 거부(denial of service attack)와 같은 것들에 취약한 잠재적 보안 위험을 줄이는 것과 같은 이유로, 또는 임의의 다른 이유로 입력들의 수를 제한하기를 원할 수 있다. 따라서, 개발자(170)는 제한된 입력들의 어레이, 예를 들어 5개의 명시적인 수치 인수들만을 허용하도록 제1 계약(110)을 변경(즉, 제2 계약(140)을 생성)할 수 있다. 현재의 웹 서비스 구현(130)이 오픈 엔디드 어레이를 갖도록 생성된 경우, 5개보다 많은 입력들은 제2 계약(140)에 따른 서비스와 더 이상 통신하지 못할 수 있다.

<35> 따라서, 실시예들은 웹 서비스 구현(130)과 제2 계약(140) 간의 변경을 자동 평가하고 구현(130)의 적어도 일부를 수정하는 데 사용되는 적응 모듈(150)을 제공한다. 이 모듈(150)은 본질적으로 제2 계약(140)으로부터 새로운 클래스 세트를 생성하고 새로운 클래스를 구현(130) 내의 기존 클래스와 비교하여 요구되는 바와 같은 변경을 행하는 차별화 시스템을 제공한다. 이러한 변경은 예를 들어 구현 골격(예를 들어, 클래스, 클래스 속성, 메소드, 메소드 시그니처, 메소드 속성, 데이터 구조 등)이 제2 계약(140)을 따르도록 보장한다. 예를 들어, 비교 모듈은 웹 서비스 구현(130); 서비스에 의해 현재 및/또는 이전에 사용된 유형 클래스(들)(135); 갱신된 또는 제2의 서비스 계약(140); 및/또는 최초 또는 제1 서비스 계약(110) 중 하나 이상을 입력으로 취할 수 있다. 최초 구현(130)을 제2 계약(140)에 따르도록 하려는 시도로서, 구현(130)(및/또는 경우에 따라 제1 계약(110))과 제2 계약(140) 간에 식별되는 변경에 기초하여, 웹 서비스 구현(130)의 하나 이상의 부분을 수정하여, 웹 서비스 구현(160)을 생성할 수 있다.

<36> 실시예들은 구현(130) 내의 메소드들의 부분들(예를 들어, 비지니스 로직)이 통상적으로 수정된 구현(160)에서 변경되지 않은 채로 유지될 것을 규정한다는 점에 유의한다. 예를 들어, 후술하는 바와 같이, 하나의 동작의 제거는 대응 메소드가 코멘트 아웃(commenting out)되거나 플래그(flagged)되거나, 이동되는 것 등을 유발할 수 있다. 따라서, 수정된 구현(160)은 컴파일 또는 실행되지 못할 수 있다. 따라서, 프로세스를 완료하기 위하여, 개발자(170)는 비지니스 로직 입력 모듈(165)(또는 다른 모듈)을 사용하여, (1) 새로운 계약에는 존재하지만 이전 계약에는 존재하지 않는 동작들에 대응하는 추가된 메소드들에 대한 적절한 비지니스 로직 및/또는 메소드 본문 코드를 제공하는 것, (2) 시그니처가 변경되었고 새로운 또는 변경된 거동을 요구하고 그리고/또는 시그니처 내의 새로운 그리고/또는 변경된 유형들을 참조하는 메소드들에 대해 요구되는 바와 같이 비지니스 로직 및/또는 메소드 본문 코드를 수정하는 것, 또는 (3) 이전에 존재하는 동작들이 계약으로부터 제거된 결과로서 더 이상 동작들로 맵핑되지 않는 메소드들을 옵션으로 제거하는 것 중 하나 이상을 행하는 것이 필요할 수 있다.

<37> 수정된 구현(160)에 대한 비지니스 로직이 필요한지의 여부와 관계없이, 제2 계약(140)에 따르도록 서비스 구현(130)을 수정하는 데 사용되는 입력들은 변할 수 있다. 예를 들어, 적응 모듈(150)은 통상적으로 웹 서비스 구현(130)을 입력으로 취할 필요가 있을 것이다. 그러나, 다른 입력들도 변경이 어떻게 식별되는지에 따라 변할 수 있다. 예를 들어, 제2 서비스 계약(140)으로부터의 유형 클래스(들)(135)가 식별되어, 웹 서비스 구현(130) 내의 클래스와 직접 비교될 수 있다. 대안으로, 또는 추가적으로, 제2 서비스 계약(140)의 골격 코드(145)가 골격 모듈(142)을 이용하여 생성된 후, 제1 계약(110)에 대한 골격 코드(115)와 비교되어 그에 대한 변경이 식

별될 수 있다.

<38> 물론, 웹 서비스 구현(130)과 제2 서비스 계약(140) 간의 변경을 식별하기 위한 다른 입력들도 이용 가능하며, 본 명세서에서 고려된다. 예를 들어, 후술하는 바와 같이, 사용자 입력 또는 다른 파라미터들이 수정된 서비스 구현(160)을 완성하는 데 필요한 다양한 사양을 결정하는 데 필요한 사례들이 존재할 수 있다. 따라서, 도 1A에 도시된 바와 같이, 실시예들은 개발자(170)(또는 입력 모듈(147) 내의 몇몇 자동화된 메커니즘)가 수정된 서비스 구현(160)을 생성하기 위해 적응 모듈(150)에 의해 사용되는 다른 파라미터들을 정의(예를 들어, 특정 결합의 맵핑을 지정)하는 것을 허가하는 입력 모듈(147)을 제공한다. 따라서, 구현(130)과 제2 계약(140) 간의 변경을 식별하기 위해 적응 모듈(150)에 의해 사용되는 임의의 특정 입력들은 본 명세서에서 단지 예시적인 목적으로만 사용된다.

<39> 또한, 구현(130)의 대부분의 임의 부분이 제2 서비스 계약(140)에 따르도록 자동으로 수정 또는 갱신될 수 있다는 점에 유의한다. 예를 들어, 클래스 속성들은 변경되지 않을 수 있지만, 메소드, 메소드 시그니처, 메소드 속성 또는 다른 특성들은 변경될 수 있다. 따라서, 실시예들은 구현 골격(즉, 클래스 유형, 클래스 속성, 메소드, 메소드 속성, 메소드 시그니처(명칭, 유형, 파라미터, 비지니스 로직 등을 포함함), 데이터 구조 또는 다른 특성들)에 대해 발생하는 대부분의 임의의 변경을 식별하도록 구성된다. 따라서, 수정된 구현(130)의 임의의 특정 부분은 단지 예시적으로 사용되며, 명시적으로 청구되지 않는 한은 여기에 설명되는 실시예들의 범위를 제한 또는 축소하는 것을 의도하지 않는다.

<40> 전술한 바와 같이, 여기에 설명되는 실시예들은 귀중한 정보를 포함하거나 사용자가 소정의 다른 상황에서 사용하기를 원할 수 있는 상당한 노력의 결과를 나타낼 수 있는 갱신 또는 수정된 구현(160) 내의 사용자가 작성한 비지니스 로직과 같은 메소드(들)의 부분들의 유지를 가능하게 한다. 예를 들어, 구현(130)의 변경이 동작의 재명명으로부터 발생하는 경우, 유지되는 메소드 본문 코드는 새로 명명된 메소드로 이동되기만 하면 된다. 메소드의 비지니스 로직 또는 다른 부분들의 유지는 임의 수의 방법으로 이루어질 수 있다는 점에 유의한다. 예를 들어, 변경된 동작의 명칭은 대응 메소드의 재명명을 유발하지 않을 수도 있다. 또한, 제1 계약(110)에서 하나의 동작의 취소는 코멘팅 아웃, 속성들의 제거, 또는 메소드가 웹 서비스의 동작으로서 노출되지 않아야 한다는 것을 의미하는 다른 마크들, 아니면 이전 메소드가 제거되지 않도록 하기 위한 이전 메소드 및 그가 포함하는 비지니스 로직의 마킹을 유발할 수 있다.

<41> 게다가, 다른 실시예들은 또한 수정된 구현(160)을 생성하는 프로세스 동안 코멘트들의 삽입을 허가한다. 예를 들어, 코멘트들은 어느 일시에 변경이 발생하였는지 그리고/또는 무엇이 변경을 발생시켰는지와 같은 정보를 포함할 수 있다. 물론, 다른 정보도 제공될 수 있다. 그럼에도, 추가 코멘트들은 또한 개발자(170)가 전반적인 서비스 갱신 활동을 완료하는 데 필요한 작업을 식별하는 것을 돕는다.

<42> 여기에 설명되는 실시예들을 더 완전히 이해하기 위하여, 도 3은 여기에 설명되는 실시예들을 실시하는 데 사용될 수 있는 특정 시스템의 흐름도를 제공한다. 특정 조회들의 결정은 도면에 도시된 것과 다른 순서로 발생할 수 있다는 점에 유의한다. 게다가, 사용되는 특정 조회들은 또한 단지 예시적이며, 여기에 설명되는 실시예들의 범위를 제한 또는 축소하고자 하는 의도는 없다.

<43> 도 3에 도시된 특정 로직은 기본적으로 차별화 시스템인데, 이는 서비스 계약으로부터 새로운 클래스 세트를 생성한 후, 새로운 클래스들을 구현 내의 기존 클래스들과 비교하여 요구되는 바와 같은 변경을 행한다. 흐름은 서비스 구현에 대한 변경들의 다양한 양태들을 차례로 처리한다. 예를 들어, 먼저 흐름은 웹 서비스 클래스 상에서 속성들을 수정하여, 갱신된 서비스 계약 내의 웹 서비스 사양들을 따르게 한다. 이어서, 흐름은 각각의 메시지 결합(예를 들어, SOAP)에 의해 참조되는 각각의 동작을 처리하고, 웹 메소드 속성과 함께 존재하는 매칭 메소드가 있음을 보장하며, 모든 다른 메소드로부터 웹 메소드 속성을 제거한다. 이어서, 흐름은 메시지 정의들에서 참조되는 스키마 유형들(예를 들어, XML)을 처리하고, 선택 가능 사용자 옵션으로서, 데이터 직렬화 클래스들의 교체 세트를 생성하거나 기존 클래스들을 갱신한다.

<44> 도시된 바와 같이, 시스템은 먼저 웹 서비스 구현의 웹 서비스 결합 명칭 및/또는 결합 명칭 공간이 제2/갱신된 웹 서비스 계약과 다른지를 판정한다(판정 블록 302). 웹 서비스 결합 명칭 및/또는 결합 명칭 공간이 다른 경우(판정 블록 302에서 예), 시스템은 웹 서비스 결합 명칭 및/또는 결합 명칭 공간을 자동으로 갱신할 것이다(304). 그러나, 개발자가 결합 명칭 및 명칭 공간을 유지하더라도 상이한 서비스 명칭하에 거동을 제공하기로 결정했을 수도 있으므로, 통상적으로 서비스 명칭/명칭 공간은 갱신되지 않아야 한다는 점에 유의한다. 거동 유형을 정의하고 제2/갱신된 웹 서비스 계약과의 매칭에 사용되는 것은 서비스 명칭이 아니라 결합 명칭 및 명칭 공간이다.

<45> 예를 들어, 결합 명칭 및 결합 명칭 공간이 WebService1 & http://tempuri.org에서 GolfCatalog & http://catalog.org로 변경된 것으로 가정한다. 아래의 코드 수정은 웹 서비스 구현 내에서 자동으로 이루어질 수 있다.

<46> 제2 웹 서비스 계약을 따르기 전:

```
[System.Web.Services.WebServiceBinding(Name = "WebService1")]
public class WebService1 : System.Web.Services.WebService
```

<47>

<48> 제2 웹 서비스 계약 문서를 자동으로 따른 후:

```
[System.Web.Services.WebServiceBinding(Name = "GolfCatalog",
Namespace = "http://catalog.com")]
public class WebService1 : System.Web.Services.WebService
```

<49>

<50> 상기 예에서, 서비스 명칭은 결합 명칭 및 명칭 공간의 갱신과 함께 변경되지 않았음에 유의한다.

<51> 웹 서비스 결합 명칭 및 명칭 공간의 갱신 후, 시스템은 기존의 결합 명칭을 포함하는 웹 서비스 구현의 메소드들이 새로운 결합 명칭을 포함하도록 자동으로 갱신할 수 있다(306). 예를 들어, 아래의 의사 코드 수정은 결합 명칭을 "WebService1"에서 "GolfCatalog"로 변경한 후에 웹 서비스 구현에서 자동으로 이루어질 수 있다.

<52> 제2 웹 서비스 계약을 따르기 전:

```
[System.Web.Services.WebMethod(...)]
[System.Web.Services.Protocols.SoapDocumentMethod(Binding =
"WebService1")]
public void getCatalog()
```

<53>

<54> 제2 웹 서비스 계약 문서를 자동으로 따른 후:

```
[System.Web.Services.WebMethod(...)]
[System.Web.Services.Protocols.SoapDocumentMethod(Binding =
"GolfCatalog")]
public void getCatalog()
```

<55>

<56> 새로운 결합 명칭으로 메소드들을 갱신한 후, 또는 웹 서비스 결합 명칭 또는 결합 명칭 공간이 제2 서비스 계약과 다르지 않은 경우(판정 블록 302에서 아니오), 시스템은 웹 서비스 계약 내의 동작 명칭에 매칭되는 웹 서비스 구현 내의 각각의 메소드에 대해 메소드 시그니처들 및 속성들이 제2/갱신된 서비스 계약 내의 동작들을 따르도록 갱신한다. 이어서, 시스템은, 제2/갱신된 계약 내의 각각의 동작에 대해, 웹 서비스 구현 메소드(들)와 제2/갱신된 서비스 계약 동작(들)을 비교한다(308). 예를 들어, 메소드 명칭들은 동작 명칭들과 비교될 수 있는데, 이는 후술하는 바와 같이 일대일 결합 상관을 가정한다. 매칭이 존재하는 경우(판정 블록 316에서 예), 시스템은 메소드가 웹 서비스 동작으로서 노출되는지를 판정한다(판정 블록 310). 웹 서비스 동작들로서 노출되지 않은 하나 이상의 메소드가 존재하는 경우(판정 블록 310에서 아니오), 비노출 메소드에 웹 메소드 속성(들)을 추가함으로써 메소드들이 자동으로 노출된다(312). 이것은 예를 들어, 웹 메소드 속성들이 정의되는 인터페이스들을 사용하고 있고 클래스 자체 상에서 웹 메소드 속성들을 정의하지 않은 사용자들에 대한 업그레이드 시나리오를 제공한다.

<57> 이와 달리, 매칭 메소드가 노출되는 경우(판정 블록 310에서 예 및/또는 블록 312에서 새로 노출된 서비스 메소드들로부터), 노출된 웹 메소드들의 구현 메소드 시그니처들 및 속성들이 제2/갱신된 웹 서비스 계약을 따르도록 자동 갱신된다(318). 시스템은 메소드/메소드 명칭 및 웹 메소드 속성을 변경하지 않을 수도 있다는 점에

유의한다.

<58> 예를 들어, 제2/갱신된 웹 서비스 계약이 getCatalog 동작이 현재 ProductId를 파라미터로서 취하고 System.Xml.Xmldocument를 반환한다고 기술하고 있는 경우, 아래의 갱신들이 자동으로 이루어질 수 있다.

<59> 예 1

<60> 제2 웹 서비스 계약에 따르기 전:

```
[System.Web.Services.WebMethod(EnableSession = true)]
[SoapDocumentMethod(...)]
public void getCatalog()
```

<61>

<62> 제2 웹 서비스 계약 문서를 자동으로 따른 후:

```
[System.Web.Services.WebMethod(EnableSession = true)]
[SoapDocumentMethod(...)]
public System.Xml.XmlDocument getCatalog(String ProductId)
```

<63>

<64> 예 2

<65> 제2 웹 서비스 계약을 따르기 전:

```
[System.Web.Services.WebMethod(MessageName = "getCatalog", ,
EnableSession = true)]
[SoapDocumentMethod(...)]
public void getGolfCatalog()
```

<66>

<67> 제2 웹 서비스 계약 문서를 자동으로 따른 후:

```
[System.Web.Services.WebMethod(MessageName = "getCatalog", ,
EnableSession = true)]
[SoapDocumentMethod(...)]
public System.Xml.XmlDocument getGolfCatalog(String ProductId)
```

<68>

<69> 시스템은 또한 웹 서비스 구현 내의 갱신 속성들을 자동으로 갱신할 수 있다. 예를 들어, 제2/갱신된 웹 서비스 계약 문서가 본문 스타일이 문서가 아니라 RPC이어야 한다고 기술하고 있는 경우, 아래의 갱신이 이루어질 수 있다.

<70> 제2 웹 서비스 계약을 따르기 전:

```
[System.Web.Services.WebMethod(EnableSession = true)]
[System.Web.Services.Protocols.SoapDocumentMethod(Binding =
"WebService1")]
public void getCatalog()
```

<71>

<72> 제2 웹 서비스 계약 문서를 자동으로 따른 후:

```
[System.Web.Services.WebMethod(EnableSession = true)]
[System.Web.Services.Protocols.SoapRpcMethod(Binding =
"WebService1")]
public void getCatalog()
```

<73>

<74> 메소드 시그니처들은 제2/갱신된 웹 서비스 계약 문서를 따르도록 수정되고 있을 때 또한 변경되는 문자열 또는 정수와 같은 프리미티브(primitive) 메시지 유형들을 포함할 수 있다는 점에 유의한다. 구현을 갱신하는 한 가지 방법은 단순히 프리미티브 유형을 교체하는 것일 수 있다. 그러나, 개발자가 프리미티브 유형의 고유 기능을 유지하기를 원할 수 있는 사례들이 존재한다. 따라서, 실시예들은 시스템이 메소드를 노출되지 않은 채로 유지하기 위해 직렬화(serialization)의 일부로서 포함되지 않을 메시지에 속성을 추가하는 것을 가능하게 한다. 이것은 기능이 제거되지 않고, 수정된 웹 서비스 구현에서 구현되지 않는 동안에 다른 개발자가 이용할 수 있는 것을 보장하는 것을 돕는다.

<75> 판정 블록 316으로 돌아가서, 제2/갱신된 계약 내의 동작 명칭이 메소드 명칭과 매칭되지 않는 경우(판정 블록 316에서 아니오), 시스템은 (블록 326에 도시된 바와 같이) 수정된 서비스 구현에 시그니처들을 가진 웹 메소드(들)를 추가함으로써 나머지 메소드들(즉, 갱신된 계약 내의 동작들 및/또는 최초 구현 내의 메소드들)을 자동으로 노출시킨다.

<76> 수정된 구현 내의 모든 메소드가 노출되면, 수정된 구현 내의 각각의 메소드에 대해, 시스템은 노출된 메소드(들)가 수정된 웹 서비스 구현 내에 존재하지만 제2/갱신된 웹 서비스 계약 문서 내에 존재하지 않는지를 판정한다(판정 블록 322). 노출된 메소드(들)가 수정된 웹 서비스 구현 내에, 그리고 제2/갱신된 계약 내에 존재하는 경우(판정 블록 322에서 아니오), 시스템은 종료된다(325). 이와 달리, 노출된 동작이 수정된 웹 서비스 구현에는 존재하지만, 제2/갱신된 웹 서비스 계약 문서에는 존재하지 않는 경우(판정 블록 322에서 예), 웹 메소드의 적어도 일부가 유지된다. 예를 들어, 메소드가 더 이상 웹 서비스 동작으로서 노출되지 않도록, 메소드, 메소드 속성, 메소드 시그니처, 또는 다른 데이터 구조가 자동으로 코멘트 아웃될 수 있다. 대안으로, 메소드, 메소드 속성, 메소드 시그니처 등이 제거되거나 후속 참조를 위해 다른 곳에 저장될 수 있다. 또한, 시스템은 메소드가 언제 그리고 왜 노출되지 않는지를 기술하는 추가 코멘트를 추가할 수 있다. 이것은 대응 메소드 시그니처들, 파라미터들 및/또는 임의의 특정 비지니스 로직을 갖는 웹 메소드가 웹 서비스 구현으로부터 제거되지 않지만, 단순히 컴파일되지 않을 것임을 보장한다.

<77> 예를 들어, getCatalog 동작이 자동으로 코멘트 아웃된 경우, 아래의 수정이 이루어질 수 있다.

<78> 제2 웹 서비스 계약을 따르기 전:

```
[System.Web.Services.WebMethod(MessageName = "Catalog",
EnableSession = true),
System.Web.Services.Protocols.SoapDocumentMethod(Binding =
"WebService1")]
public void getCatalog()
```

<79>

<80> 제2 웹 서비스 계약 문서를 자동으로 따른 후:

```
/// [7/2/2004 4:10 PM] The WebMethod attribute was removed as this
operation no longer conforms to the WSDL.
/// [System.Web.Services.WebMethod(MessageName = "Catalog",
EnableSession = true),
[System.Web.Services.Protocols.SoapDocumentMethod(Binding =
"WebService1")]
```

<81> public void getCatalog()

<82> 몇몇 실시예에서, 이후 개발자는 수정된 웹 서비스 구현을 검사하고 추가 수정을 행할 수 있다. 예를 들어, 수정된 웹 서비스가 컴파일 또는 실행될 때, 하나 이상의 에러가 발생할 수 있다. 에러를 치료하기 위하여, 개발자는 전술한 바와 같이 구현을 수정할 수 있다.

<83> 전술한 바와 같이, 상기 시스템은 수정된 구현에서 어느 메소드들이 노출되어야 하는지를 더 정의하기 위하여 사용자 또는 다른 자동화된 메커니즘으로부터의 추가 입력 파라미터들을 필요로 할 수 있다. 예를 들어, 전술한 시스템은 구현과 계약 간의 결합들의 단일 특정 맵핑 쌍, 즉 일대일 상관을 가정하였다. 그러나, 계약들(예를 들어, WSDL 문서들)은 계약이 구현으로 맵핑될 때 다수의 결합(각각의 결합은 동작들의 개별 세트를 정의함)을 포함할 수 있다. 예를 들어, 계약 내에서 다수의 버전이 지원되는 사례들이 존재할 수 있으며, 따라서 명칭 또는 명칭 공간은 예를 들어 버전 식별자를 포함할 수 있다(이러한 명칭 및/또는 명칭 공간에 대한 변경은 결합 내의 임의의 동작의 거동 또는 시그니처에 대해 임의의 자료 변경이 행해질 때 전형적이다).

<84> 따라서, 동일 명칭의 동작들은 상이한 결합들 상에서 발생할 수 있고, 따라서 동작 명칭 맵핑만으로는 불충분할 수 있는데, 이것은, 계약 내에 있고 서로 관련될 수 있도록 구현 코드 내에 속성들로서 기록되는 명칭 및 명칭 공간 특성들에 의해 결합이 충분히 식별되기 때문이다. 동작들(결합들)의 다수의 세트가 계약 내에 정의되는 사례에서, 이들 세트의 명칭들의 일부(즉, 명칭 또는 명칭 공간)만이 제1 및 제2 계약 사이에서 변경된 경우에 시스템은 추가 입력을 요구할 수 있다. 이러한 입력은 구현 수정의 실행 동안의 직접 사용자 입력의 형태로, 또는 맵핑을 지정할 수 있는 실행에 대한 파라미터들로서 제공될 수 있다.

<85> 즉, 전술한 시스템은 물론, 후술하는 실시예들은 다음과 같은 상황, 즉 (1) 각기 단일 결합(즉, 일대일 상관)을 갖는 계약 및 구현-이 경우, 계약 내의 결합 및 그의 동작들은 명칭들 및/또는 명칭 공간들에 관계없이 구현을 따르게 하는 데 사용됨-의 상황; (2) 발생할 적응을 위해 계약 및/또는 구현이 다수의 결합을 갖고, 명칭 및 명칭 공간이 정확히 매칭되어야 하는 상황(구현 내에 없는 계약 내의 임의의 결합들이 구현에 추가될 수 있거나, 계약 내에 없는 구현 내의 임의의 결합이 예를 들어 이를 코멘트 아웃함으로써 제거되어야 한다); 및 (3) 계약 내의 어떠한 결합들이 구현 내의 어떠한 결합들로 맵핑되어야 하는지를 명확하게 식별하는 맵핑 명령들이 (예를 들어, 사용자 입력 또는 다른 자동화된 방식을 통해) 제공된 상황(맵핑되지 않은 임의의 결합들은 무시되거나 (2)에서와 같이 처리될 수 있다)에서 실행되는 것이 필요할 수 있다. 그러나, 맵핑된 결합 및 맵핑되지 않은 결합이 처리되는 방법은 본 명세서에서 예시적인 목적으로 사용되며, 명시적으로 청구되지 않는 한은 여기에 설명되는 실시예들의 범위를 제한 또는 축소하지 않아야 한다는 점에 유의한다.

<86> 여기에 설명되는 실시예들은 기능 단계들 및/또는 비 기능 동작들을 포함하는 방법들과 관련하여 설명될 수도 있다. 다음 섹션들 중 일부는 본 발명의 실시에 있어서 수행될 수 있는 단계들 및/또는 동작들의 설명을 제공한다. 일반적으로, 기능 단계들은 달성되는 결과들과 관련하여 발명을 설명하는 반면, 비 기능 동작들은 특정 결과를 달성하기 위한 보다 구체적인 동작들을 설명한다. 기능 단계들 및/또는 비 기능 동작들이 특정 순서로 설명 또는 청구될 수 있지만, 본 발명은 단계들 및/또는 동작들의 임의의 특정 순서 또는 조합으로 제한될 필요는 없다. 또한, 청구항들의 기재에서-그리고 도 2의 흐름도에 대한 아래의 설명에서- 단계들 및/또는 동작들의 사용은 이러한 용어들의 원하는 특정 사용을 지시하기 위해 사용된다.

<87> 전술한 바와 같이, 도 2는 본 발명의 다양한 실시예의 흐름도를 나타낸다. 아래의 도 2에 대한 설명은 종종 도 1A 및 1B 및 3의 대응 요소들을 참조할 것이다. 도면들의 특정 요소를 참조할 수 있지만, 이러한 참조는 예시적인 목적으로 사용되며, 명시적으로 청구되지 않는 한은 설명되는 실시예들의 범위를 제한 또는 축소하는 것을 의도하지 않는다.

<88> 이제, 도 2를 참조하면, 웹 서비스 구현을 웹 서비스 계약의 변경에 따르도록 자동으로 수정하기 위한 방법(200)의 흐름도가 도시되어 있다. 방법(200)은 웹 서비스 구현의 일부를 자동으로 수정하는 단계(210)를 포함한다. 예를 들어, 적응 모듈(150)을 이용하여 웹 서비스 구현(130)의 일부를 자동으로 수정할 수 있다. 통상적으로, 이러한 수정은 그 안에 포함된 메소드들의 하나 이상의 부분(예를 들어, 비지니스 로직)을 유지할 것이다. 또한, 이러한 수정은, 웹 서비스 구현 결합 명칭 또는 명칭 공간 중 어느 하나가 제2 웹 서비스 계약(140)과 다른 경우에 이를 갱신하는 단계(304); 제2 웹 서비스 계약 문서(140)에 포함되지 않은 동작에 대한 웹 메소드를 유지하는 단계(324), 제2 웹 서비스 계약에는 포함되어 있지만 웹 서비스 구현에는 포함되지 않은 동작들에 대한 메소드 시그니처 및 속성들을 갖는 메소드를 추가하는 단계(326); 동작에 매칭되는 웹 메소드들에 대한 메소드 시그니처를 수정하는 단계(318); 메소드가 동작으로서 노출되도록 메소드에 속성을 추가하는 단계; 및/또는 메소드가 더 이상 동작으로서 노출되지 않지만, 메소드의 내용이 유지되도록 메소드에 대한 속성을 제거하는 단계 중 하나 이상을 포함할 수 있다.

<89> 단계 210은 제1 웹 서비스 계약 문서에 따라 구축된 웹 서비스 구현을 수신하는 동작(201)을 포함한다. 예를 들어, 웹 서비스 구현(130)이 적응 모듈(150)에 의해 수신될 수 있다. 웹 서비스 구현은 제1 서비스 계약 문서(110)에 따라 구축될 수도 있다. 전술한 바와 같이, 제1 계약(110)은 통상적으로 제공되는 하나 이상의 서비스의 거동을 기술할 것이다. 따라서, 제1 기술(110)은 특히, 제공되는 동작(들), 교환되는 메시지들의 포맷은 물론, 웹 서비스 구현(130)이 컴퓨팅 시스템(100)의 다른 서비스들과 통신하는 방법을 정의하는 다른 특성들을 기술한다.

<90> 단계 210은 제1 웹 서비스 계약 문서에 대한 변경(들)을 정의하는 제2 웹 서비스 계약 문서를 수신하는 동작(202)을 더 포함한다. 예를 들어, 적응 모듈(150)은 제1 서비스 계약(110)의 부분들에 대한 변경들을 포함할 수 있는 제2 웹 서비스 계약(140)을 수신할 수 있다. 이러한 변경들은 제공되는 동작(들), 교환되는 메시지들의 포맷은 물론, 웹 서비스 구현(130)이 컴퓨팅 시스템(100)의 다른 서비스들과 통신하는 방법을 정의하는 다른 특성들을 변경할 수 있다. 변경들은 웹 서비스 구현(130)이 아마도 제2 웹 서비스 계약(140)을 구현하는 서비스를 기대하는 엔드포인트(예를 들어, 클라이언트)와 통신할 수 없도록 웹 서비스 구현(130)의 거동에 영향을 미친다는 점에 유의한다. 또한, 제1 계약(110) 및 제2 계약(140) 양자는 WSDL과 같은 소정의 NPDL 형태일 수 있다는 점에 유의한다.

<91> 단계 210은 웹 서비스 구현과 제2 웹 서비스 계약 간의 변경(들)을 식별하는 동작(203)을 더 포함한다. 예를 들어, 적응 모듈(150) 또는 컴퓨팅 시스템(100)의 소정의 다른 컴포넌트 및/또는 모듈은 웹 서비스 구현(130)과 제2 웹 서비스 계약(140) 간의 변경(들)을 식별할 수 있다. 몇몇 실시예에서, 기존 웹 서비스 구현(예를 들어, 웹 서비스 구현(130))의 부분들에 대한 변경들은 서비스 구현과 제2 웹 서비스 계약 문서(예를 들어, 제2 서비스 계약 문서(140))의 부분(들)을 비교함으로써 식별된다. 물론, 전술한 바와 같이 변경들을 식별하는 다른 방법들도 본 명세서에서 고려된다. 예를 들어, 다른 실시예들은 제2 웹 서비스 계약(140)에 기초하여 생성되는 골격 코드(145)를 사용하고, 이를 웹 서비스 구현(130) 또는 제1 웹 서비스 계약(110)과 비교하여 그에 대한 변경들을 식별할 수 있다. 어느 경우에나, 변경들이 식별되면, 적응 모듈(150)은 전술한 바와 같이 수정된 웹 서비스 구현(160)을 생성할 수 있다.

<92> 비지니스 로직 입력 인터페이스 또는 모듈(165)을 이용하여 웹 서비스 구현(160)의 수정 부분을 노출시키기 위한 다른 실시예들이 더 제공된다. 이것은 개발자(170)가 제2 웹 서비스 계약 문서(140)에는 존재하지만 제1 웹 서비스 계약 문서(110)에는 존재하지 않는 동작들에 대응하는 웹 서비스 구현(130)에 추가된 메소드들에 대한 적절한 비지니스 로직 코드를 제공하는 단계; 시그니처들 또는 메시지 유형들이 제2 웹 서비스 계약 문서(140)에 따르도록 수정된 메소드들에 대한 기존 비지니스 로직 코드를 수정하는 단계; 및 동작들이 제1 웹 서비스 계약 문서(110)에는 존재하지만, 제2 웹 서비스 계약 문서(140)에는 존재하지 않는 결과로서 더 이상 동작들로 맵핑되지 않는 웹 메소드들을 제거하는 단계 중 일부를 수행하는 것을 허가한다.

<93> 본 발명은 그의 사상 또는 본질적인 특징으로부터 벗어나지 않고 다른 특정 형태들로 구현될 수 있다. 설명되는 실시예들은 모든 점에서 제한적이 아니라 단지 예시적인 것으로 간주되어야 한다. 따라서, 본 발명의 범위는 위의 설명이 아니라 첨부된 청구범위에 의해 지시된다. 청구범위의 균등물의 의미 및 범위 내에 있는 모든 변경은 본 발명의 범위 내에 포함되어야 한다.

도면의 간단한 설명

<9> 도 1A는 실시예들에 따라 웹 서비스 계약 문서의 변경에 응답하여 웹 서비스 구현을 자동으로 수정하기 위한 분

산형 컴퓨팅 시스템을 나타내는 도면.

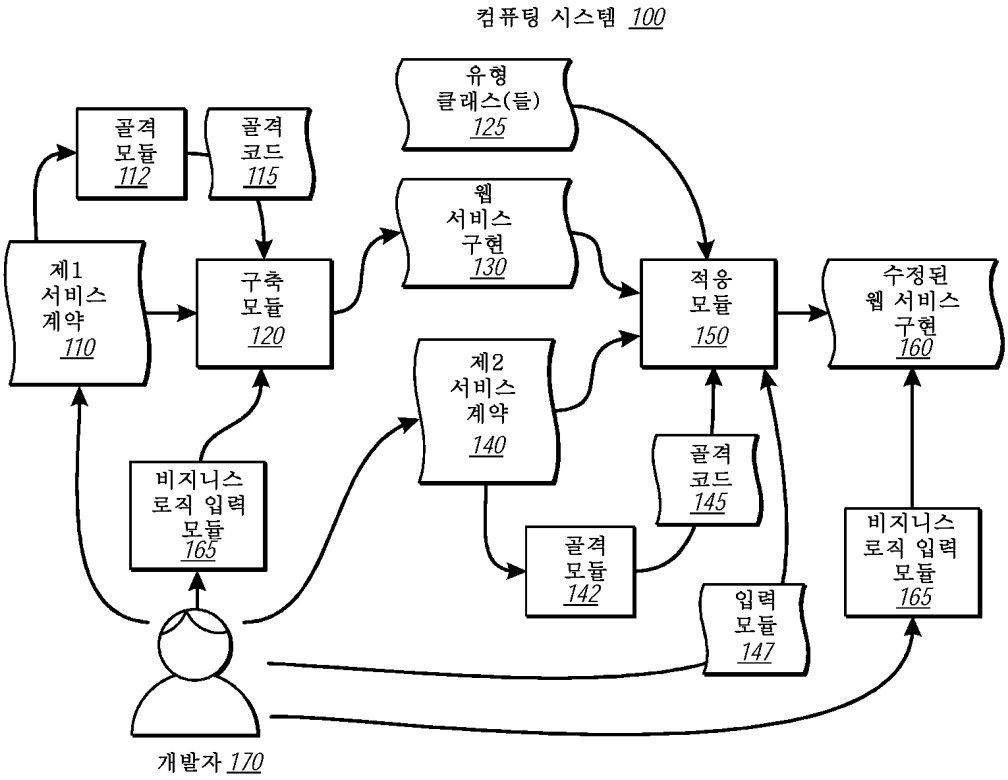
<10> 도 1B는 서비스 구현의 일례를 나타내는 도면.

<11> 도 2는 실시예들에 따라 웹 서비스 계약 문서의 변경에 응답하여 서비스 구현을 자동으로 수정하기 위한 방법의 흐름도.

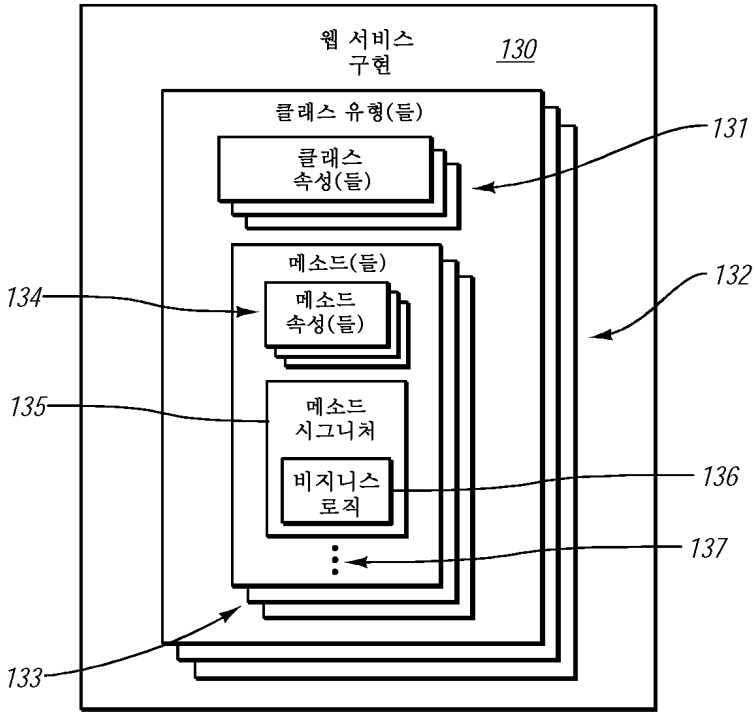
<12> 도 3은 실시예들에 따라 웹 서비스 계약 문서의 변경에 응답하여 서비스 구현을 자동으로 수정하는 특정 시스템의 상세 흐름도.

도면

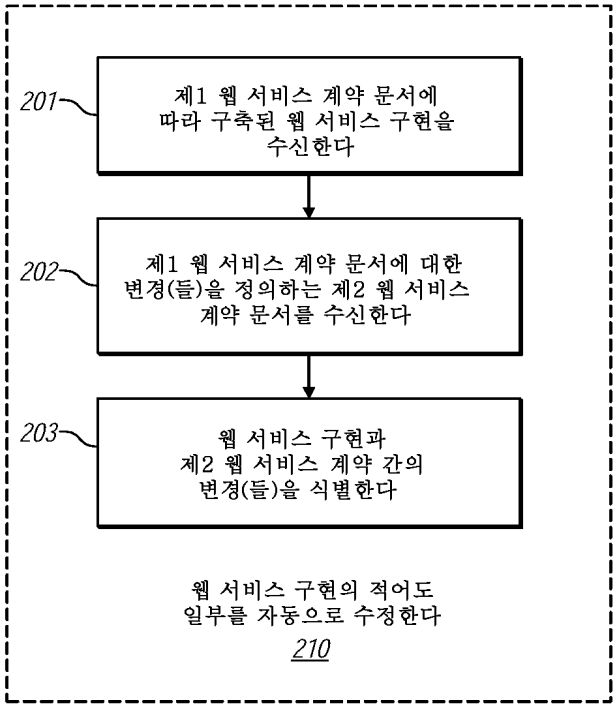
도면1A



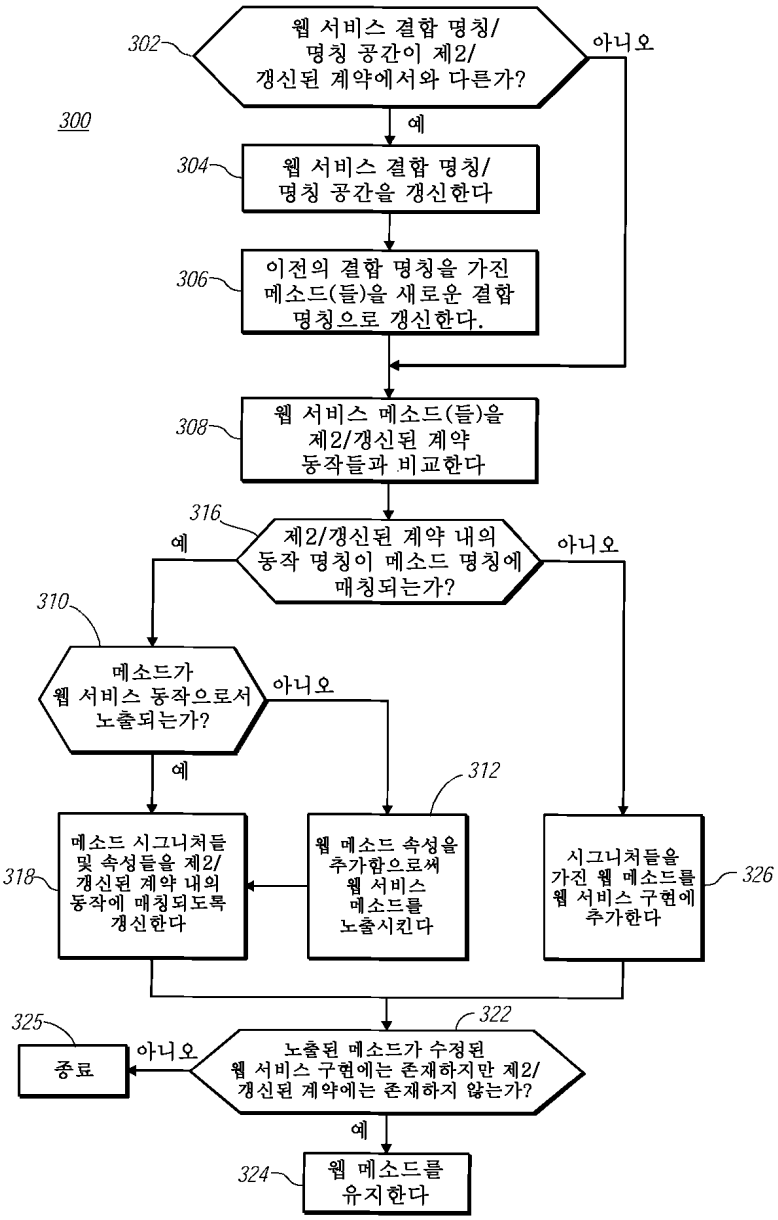
도면1B


웹 서비스
구현
130
클래스 유형(들)
클래스
속성(들)
131
메소드(들)
메소드
속성(들)
132
134
135
메소드
시그니처
비지니스
로직
136
137
133

도면2


201
제1 웹 서비스 계약 문서에
따라 구축된 웹 서비스 구현을
수신한다
202
제1 웹 서비스 계약 문서에 대한
변경(들)을 정의하는 제2 웹 서비스
계약 문서를 수신한다
203
웹 서비스 구현과
제2 웹 서비스 계약 간의
변경(들)을 식별한다
웹 서비스 구현의 적어도
일부를 자동으로 수정한다
210

도면3


300
302
웹 서비스 결합 명칭/
명칭 공간이 제2/
갱신된 계약에서와 다른가?
아니오
예
304
웹 서비스 결합 명칭/
명칭 공간을 갱신한다
306
이전의 결합 명칭을 가진
메소드(들)을 새로운 결합
명칭으로 갱신한다.
308
웹 서비스 메소드(들)을
제2/갱신된 계약
동작들과 비교한다
316
제2/갱신된 계약 내의
동작 명칭이 메소드 명칭에
매칭되는가?
예
아니오
310
메소드가
웹 서비스 동작으로서
노출되는가?
아니오
예
312
메소드 시그니처들
및 속성들을 제2/
갱신된 계약 내의
동작에 매칭되도록
갱신한다
318
웹 메소드 속성을
추가함으로써
웹 서비스
메소드를
노출시킨다
시그니처들을
가진 웹 메소드를
웹 서비스 구현에
추가한다
326
325
종료
아니오
322
노출된 메소드가 수정된
웹 서비스 구현에는 존재하지만 제2/
갱신된 계약에는 존재하지 않는가?
예
324
웹 메소드를
유지한다