(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년09월24일
(11) 등록번호 10-1310988
(24) 등록일자 2013년09월13일

(51) 국제특허분류(Int. Cl.)
G06F 17/21 (2006.01)
(21) 출원번호 10-2008-7005679
(22) 출원일자(국제) 2006년09월07일
심사청구일자 2011년08월29일
(85) 번역문제출일자 2008년03월07일
(65) 공개번호 10-2008-0045695
(43) 공개일자 2008년05월23일
(86) 국제출원번호 PCT/US2006/034974
(87) 국제공개번호 WO 2007/030683
국제공개일자 2007년03월15일
(30) 우선권주장
11/332,468 2006년01월13일 미국(US)
60/715,986 2005년09월09일 미국(US)
(56) 선행기술조사문헌
US20040103147 A1
US20040088647 A1
US20040205653 A1
전체 청구항 수 : 총 19 항

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
데이비스, 트리스탄 에이.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
탈레가니, 알리
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
(뒷면에 계속)
(74) 대리인
제일특허법인

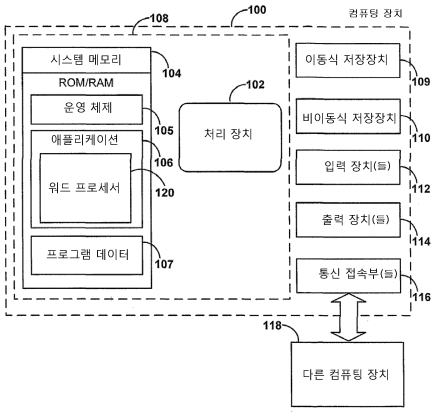
심사관 : 박상현

(54) 발명의 명칭 **애플리케이션들 사이의 XML 데이터의 실시간 동기화**

(57) 요 약

하나 이상의 데이터 저장소들이 다수의 데이터 소비자들 사이에서의 컴퓨터 생성 문서와 관련이 있는 임의의 데이터의 이용을 저장하고 관계하며 허용하기 위해 문서 내의 주 프레젠테이션 저장장치와 별도로 유지된다. 데이터 저장소는 상이한 데이터 소비자들이 하나 이상의 데이터 피스에 실시간으로 액세스하고 조작하게 하기 위해 데이터 저장소에 있는 다양한 데이터 피스들에게 API들을 노출시킨다. 다수의 데이터 소비자들은 동일한 데이터 피스에 동시에 액세스하여 편집할 수 있고, 정해진 데이터 피스에 대한 어떤 저촉 있는 변경들이든 해소된다. 각각의 데이터 소비자는 변경을 수락 또는 거절할 뿐만 아니라, 오리지널 변경의 결과로서의 추가적 부차적 변경들을 만들 수 있다. 이 방식으로, 데이터가 데이터 소비자들 사이에 실시간으로 동기화될 수 있다.

대 표 도 - 도1



(72) 발명자

존스, 브라이언, 엠.

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

사위키, 마킨

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

리틀, 로버트, 에이.

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

쿠팔라, 시라즈

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

바라크, 드라고스

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

특허청구의 범위

청구항 1

컴퓨터 생성 문서와 관련이 있고 데이터 소비자들(data consumers) 사이에 공유된 데이터를 동기화하기 위한 컴퓨터 실행 가능 명령어를 갖는 컴퓨터 판독 가능 매체에 있어서, 상기 컴퓨터 실행 가능 명령어는,

문서와 관련이 있는 XML(Extensible Markup Language)로 표현된 요소(elements)를 데이터 저장소에 저장하는 단계 -상기 데이터 저장소는 상기 문서의 프리젠테이션 뷰를 나타내는 프리젠테이션 저장소와 별개로 유지되고, 상기 문서는 하나 이상의 데이터 소비 애플리케이션에 의해 편집되며, 상기 데이터 저장소 내의 동일한 데이터 피스(a same piece of data)는 복수의 데이터 소비자에 의해 동시에 편집가능함- 와;

제1 데이터 소비자에 의한 XML 요소에 대한 변경을 시작하는 단계와;

상기 XML 요소에 대한 상기 변경에 관심이 있는 다른 데이터 소비자가 있는 지를 판정하는 단계 -상기 판정은 다른 데이터 소비자가 상기 문서와 관련된 상기 XML 요소의 변경에 대한 통지를 등록하기 위해 API(application programming interface)를 호출했는 지의 여부에 기초하고, 상기 XML 요소는 상기 데이터 저장소에 저장되며, 상기 API는 상기 다른 데이터 소비자가 상기 문서와 관련되는 다른 XML 요소의 변경에 응답하여 통지받지 않도록 지정하는 능력을 제공하고, 상기 데이터 소비자 전부는 데이터 소비 애플리케이션(data consuming application)임- 와;

상기 다른 데이터 소비자에게 상기 변경을 통지하는 단계 -상기 다른 데이터 소비자는 상기 변경의 수락, 상기 변경의 거절 및 오리지널 변경의 결과로서의 부차적 변경(side-effect change)의 시작 중 적어도 하나를 수행할 수 있음- 와;

상기 변경을 통지받은 상기 다른 데이터 소비자 전부에 의한 상기 변경의 수락에 응답하여, 상기 변경을 복수의 애플리케이션의 실행취소 스택－각각의 애플리케이션 실행 취소 스택은 각각의 데이터 소비자와 관련됨－에게 커밋(committing)하고, 상기 변경을 상기 데이터 저장소에게 커밋하는 단계와;

상기 변경을 통지받은 상기 다른 데이터 소비자 중 어느 하나에 의한 상기 부차적 변경의 시작에 응답하여, 상기 다른 데이터 소비자 각각이 상기 변경의 통지를 수신할 때까지 상기 부차적 변경이 처리되지 않도록 지연시키는 단계와;

상기 변경을 통지받은 상기 다른 데이터 소비자 중 어느 하나에 의한 변경의 거절에 응답하여, 상기 변경을 상기 복수의 애플리케이션 실행취소 스택으로부터 롤백하고, 상기 데이터 저장소를 상기 변경이 시작된 시간 전의 상태로 롤백하는 단계를 포함하는

컴퓨터 판독 가능 매체.

청구항 2

제1항에 있어서,

상기 API는 상기 문서가 열려 있는 동안에 상기 데이터 저장소 내의 상기 XML 요소에 액세스할 능력을 제공하며, 상기 API는 하나를 초과하는 데이터 소비자가 동일한 XML 요소에 동시에 액세스할 수 있도록 하는 컴퓨터 판독 가능 매체.

청구항 3

제2항에 있어서,

상기 컴퓨터 실행 가능 명령어는,

상기 제1 데이터 소비자에 의해 변경된 상기 XML 요소의 값으로 상기 다른 데이터 소비자 중 하나와 관련이 있는 디스플레이를 업데이트하는 단계를 더 포함하는 컴퓨터 판독 가능 매체.

청구항 4

제3항에 있어서,

상기 부차적 변경이 나중의 실행을 위해 대기되고(queued), 상기 나중의 실행은 상기 데이터 소비자들의 각각이

부차적 변경의 수락, 거절 및 생성 기회를 가진 후에 일어나는 컴퓨터 판독 가능 매체.

청구항 5

제3항에 있어서,

상기 변경 수락 시에 상기 변경을 커밋하는 상기 단계는 각각의 부차 효과를 처리하는 단계와 각각의 부차 효과가 수락되는지를 판정하는 단계를 포함하는 컴퓨터 판독 가능 매체.

청구항 6

제3항에 있어서,

상기 컴퓨터 실행 가능 명령어는,

상기 데이터 저장소를 상기 변경이 시작된 시간 전의 상태로 되돌리기 위해 이용될 수 있는 실행취소 목록을 유지하는 단계를 더 포함하는 컴퓨터 판독 가능 매체.

청구항 7

컴퓨터로 구현되며, 컴퓨터 생성 문서와 관련이 있는 데이터를 데이터 소비자들 사이에서 공유하기 위한 방법에 있어서,

문서와 관련이 있는 구조적 데이터 아이템(structured data item)에 대한 변경을 시작하는 단계 -상기 구조적 데이터 아이템은 XML(Extensible Markup Language)에 따라 구성되고 데이터 저장소에 저장되며, 상기 데이터 저장소는 상기 문서를 위한 프리젠테이션 저장소와 별개이며, 상기 문서는 하나 이상의 데이터 소비 애플리케이션에 의해 편집되고, 상기 데이터 저장소 내의 동일한 데이터 피스는 복수의 데이터 소비자에 의해 동시에 편집가능함- 와;

등록된 데이터 소비자에게 상기 변경의 통지를 제공하는 단계 -상기 등록된 데이터 소비자 각각은 데이터 소비 애플리케이션이고, 상기 등록된 데이터 소비자는 상기 문서와 관련된 상기 구조적 데이터 아이템 내의 변경에 대한 통지를 등록하기 위해 통지 API(application programing interface)를 호출함- 와;

나중의 처리를 위해 상기 변경의 결과인 부차적 변경을 상기 등록된 데이터 소비자 각각이 상기 변경의 통지를 수신할 때까지 대기시키는 단계와;

상기 등록된 데이터 소비자 중 하나로부터의 응답이 상기 변경의 거절인 지를 판정하고, 상기 응답이 거절인 경우, 복수의 애플리케이션 실행취소 스택으로부터 상기 변경을 롤백하고 -각각의 애플리케이션 실행취소 스택은 각각의 데이터 소비자와 관련됨-, 상기 변경 및 상기 데이터 저장소 내의 모든 데이터 소비자로부터의 어떤 부차적 변경이든 최종의 알려진 양호한 상태로 롤백하는 단계와;

상기 등록된 데이터 소비자 각각이 상기 변경을 수락하는 지를 판정하고, 상기 등록된 데이터 소비자 각각이 상기 변경을 수락할 경우, API를 통해 상기 데이터 소비자들 중 하나에 의해 제공된 XML 스키마 파일을 이용하여 상기 변경의 유효성을 검사하며, 상기 복수의 애플리케이션 실행취소 스택에게 상기 변경을 커밋하고, 상기 데이터 저장소에게 상기 변경을 커밋하는 단계를 포함하는

데이터 공유 방법.

청구항 8

제7항에 있어서,

상기 데이터 저장소 내에 포함된 구조적 데이터 아이템에 대해 상기 변경을 시작하는 단계는, 상기 데이터 소비자에게 상기 데이터 저장소 내의 상기 구조적 데이터 아이템에 액세스할 능력을 제공하는 API(application programming interface)를 노출시키는 단계를 포함하고,

상기 API는 상기 데이터 소비자들 중 하나를 초과하는 소비자가 동일한 구조적 데이터 아이템에 동시에 액세스하도록 허용하는 데이터 공유 방법.

청구항 9

제8항에 있어서,

상기 통지에 응답하여 부차적 변경을 수신하는 단계와,

상기 변경이 처리될 때까지 상기 데이터 저장소 내의 상기 부차적 변경의 실행을 지연시키는 단계를 더 포함하는 데이터 공유 방법.

청구항 10

제9항에 있어서,

상기 데이터 저장소 내의 상기 부차적 변경을 대기시키는 단계를 더 포함하는 데이터 공유 방법.

청구항 11

제9항에 있어서,

상기 등록된 데이터 소비자 각각이 상기 변경을 수락하는 지를 판정하는 단계는 상기 변경이 각각의 데이터 소비자에 의해 수락되는 지 및 상기 부차적 변경의 각각이 수락되는 지를 판정하는 단계를 포함하는 데이터 공유 방법.

청구항 12

제9항에 있어서,

상기 데이터 저장소를 상기 최종의 알려진 양호한 상태로 되돌리기 위해 이용되는 실행취소 목록을 유지하는 단계를 더 포함하는 데이터 공유 방법.

청구항 13

컴퓨터 생성 문서와 관련이 있는 데이터를 데이터 소비자들 사이에서 동기화하기 위한 시스템에 있어서,

상기 문서를 생성하고 편집하도록 구성되고 상기 문서와 관련이 있는 구조적 데이터 아이템과 상호작용하도록 구성된 내부의 데이터 소비자와, 상기 내부 데이터 소비자와 관련이 있고 상기 내부 데이터 소비자에 의해 수신된 등록된 변경을 커밋하고 롤백하도록 구성된 내부 실행취소 스택과;

상기 문서와 관련이 있으며 상기 문서를 위한 프리젠테이션 저장부와 별개로 유지되는 상기 구조적 데이터 아이템과 상호작용하도록 구성되는 외부의 데이터 소비자와, 각각이 각각의 외부 데이터 소비자와 관련되고 상기 외부 데이터 소비자에 의해 수신된 등록된 변경을 커밋하고 롤백하도록 구성된 복수의 외부 실행취소 스택과;

상기 문서와 관련이 있는 상기 구조적 데이터 아이템을 상기 문서와 별개로 저장하도록 구성된 데이터 저장소 - 상기 데이터 저장소 내의 동일한 데이터 피스는 상기 내부 데이터 소비자 및 외부 데이터 소비자에 의해 동시에 편집될 수 있음 -를 포함하되,

상기 데이터 저장소는 API 브로커를 포함하며, 상기 API 브로커는, 상기 외부의 데이터 소비자 및 상기 내부의 데이터 소비자와 상호작용하도록 구성되고, 또한 제안된 변경을 수신하고 상기 제안된 변경에 응답하여 등록된 데이터 소비자에게 상기 제안된 변경을 통지하며 상기 제안된 변경이 상기 등록된 데이터 소비자 각각에 의해 수락될 경우 상기 변경을 상기 데이터 저장소에 커밋하고 - 상기 등록된 데이터 소비자 각각은 상기 제안된 변경의 통지를 등록하기 위해 상기 API 브로커를 호출함 - 상기 제안된 변경이 하나 이상의 상기 등록된 데이터 소비자에 의해 수락되지 않을 경우에 상기 데이터 저장소가 유효한 상태에 있을 것을 보장하도록 구성되며,

상기 데이터 저장소는, 상기 제안된 변경 및 상기 제안된 변경과 관련이 있는 부차적 변경을 저장하도록 구성된 변경 저장소, 및 상기 데이터 저장소를 상기 유효한 상태로 롤백하도록 구성된 실행취소 저장소를 더 포함하고, 상기 데이터 저장소는 또한 나중의 실행을 위해 상기 데이터 소비자 각각이 상기 제안된 변경의 통지를 수신할 때까지 상기 부차적 변경을 대기하도록 구성되는

데이터 동기화 시스템.

청구항 14

제13항에 있어서,

상기 데이터 저장소 내의 상기 구조적 데이터 아이템은 XML(Extensible Markup Language)에 따라 구성되는 데이터 동기화 시스템.

청구항 15

제14항에 있어서,

상기 나중의 실행은 각각의 등록된 데이터 소비자가 부차적 변경의 수락, 거절, 및 생성 기회를 가진 후에 일어나도록 구성되는 데이터 동기화 시스템.

청구항 16

제15항에 있어서,

상기 데이터 저장소는 또한 각각의 부차 효과를 처리하고 각각의 부차 효과가 수락되는지를 판정하도록 구성되는 데이터 동기화 시스템.

청구항 17

제15항에 있어서,

상기 데이터 저장소는 또한,

상기 구조적 데이터 아이템에 대해 적용된 XML(Extensible Markup Language) 마크업 데이터에 대한 변경을 수신하고,

상기 XML 마크업 데이터에 대한 상기 변경이 안내되는 상기 구조적 데이터 아이템과 관련이 있는 XML 스키마 파일을 판독하며,

상기 XML 마크업 데이터에 대한 상기 변경이 상기 판독한 XML 스키마 파일에 따라 유효한지를 판정하고,

상기 XML 마크업 데이터에 대한 상기 변경이 상기 판독한 XML 스키마 파일에 따라 유효하지 않으면 상기 XML 마크업 데이터에 대한 상기 변경을 불허하고, 상기 XML 마크업 데이터에 대한 상기 변경이 상기 판독한 XML 스키마 파일에 따라 유효할 경우 상기 변경을 커밋하도록 구성되는 데이터 동기화 시스템.

청구항 18

제15항에 있어서,

상기 데이터 저장소는 각각이 실행취소 저장소 및 변경 저장소를 포함하는 복수의 XML 데이터 저장소를 포함하는 데이터 동기화 시스템.

청구항 19

제18항에 있어서,

각각의 상기 실행취소 저장소는 다른 실행취소 저장소와 통신하도록 구성되는 데이터 동기화 시스템.

청구항 20

삭제

명 세 서

기 술 분 야

[0001] 컴퓨터 사용자들은 그들이 쓰고, 계산하며, 조직하고, 프레젠테이션을 준비하며, 전자 메일을 보내고 수신하며, 음악을 만드는 것들을 돕는 사용자 친화적 소프트웨어 애플리케이션들에 익숙해져 있다. 예를 들어, 워드 프로세싱 애플리케이션들은 사용자들이 다양한 유용한 문서들을 준비하게 한다. 스프레드시트 애플리케이션들은 사용자들이 데이터를 입력하고 조작하며 조직하게 한다. 슬라이드 프레젠테이션 애플리케이션들은 사용자들이 텍스트, 사진, 데이터 또는 다른 유용한 개체들을 포함하는 다양한 슬라이드 프레젠테이션들을 만들게 한다.

배 경 기 술

[0002] 그러나, 그러한 애플리케이션들에 의해 만들어진 문서(예를 들어 워드 프로세싱 문서, 스프레드시트, 슬라이드 프레젠테이션 문서)들은 문서들의 컨텍스트에 의해 요구된 임의의 메타데이터의 콘텐츠들을 저장/전송하기에 제한된 편의성을 갖는다. 예를 들어, 워드 프로세싱 문서의 상단에 형성된 솔루션들은 문서의 다양한 상태, 예를 들어, 이전의 워크플로우 승인 상태(일자, 시간, 이름), 현재의 승인 상태, 완료 전 장래의 워크플로우 상태, 문서 만든 이의 이름 및 사무실 주소, 문서 변경 등을 기술하는 워크플로우 데이터의 저장을 요구할 수 있을 것이다. 이 정보를 저장하기 위한 옵션들은 제한을 갖는 문서 변수 또는 기존의 사용자 지정 OLE(object linking and embedding) 문서 속성의 이용으로 주로 제한된다. 예를 들어, 계층적 데이터가 저장되지 않을 수 있고, 문자 길이가 제한되는 것 등이 있다. 그러한 방법들을 위한 속성들은 단일의 저장소, 예를 들어, OLE 속성 저장소에 저장되며, 그것은 속성들이 저촉 가능성을 갖는 것을 의미한다. 또한, 그러한 저장된 속성들은 아무런 데이터 유효성 검사를 하지 않는다. 그러한 애플리케이션 및 관련 문서들의 사용자들이 임의의 데이터를 문서에 저장하는 것은 많은 사용자들의 보편적인 요구이지만, 어렵다.

발명의 상세한 설명

[0003] 이 개요는 아래의 발명의 상세한 설명에서 더 기술되는 개념들의 선택을 간략하게 소개하기 위해 제공된다. 이 개요는 청구 주제의 주요한 기능 또는 본질적인 기능들을 식별하려는 것도 아니고, 청구 주제의 범위를 결정함에 있어서의 보조로서 이용되려는 것도 아니다.

[0004] 하나 이상의 데이터 저장소(data store)들이 다수의 데이터 소비자(data consumer)들 사이에서 컴퓨터 생성 문서(computer-generated document)와 관련이 있는 임의의 데이터(arbitrary data)를 저장하고 관계를 가지며 이용을 허용하기 위해 문서 내의 주 프레젠테에션 저장장치(primary presentation storage)와 별도로 유지된다. 문서 메타데이터(document metadata) 등과 같은, 문서와 관련이 있는 정보를 구성하기 위한 데이터는 상이한 데이터 피스(pieces of data)들 사이의 관계가 유지된 데이터 저장소에 유지된다. 상이한 데이터 소비자들이 하나 이상의 데이터 피스에 실시간으로 액세스하고 조작하게 하기 위해 데이터 저장소가 데이터 저장소 내의 다양한 데이터 피스들에게 API(application programming interface)들을 노출시킨다. 다수의 데이터 소비자들이 동일한 데이터 피스에 동시에 액세스하고 편집할 수 있을 것이고, 정해진 데이터 피스(given piece of data)에 대한 어떤 저촉되는 변경(conflicting change)들이든 해결된다. 각각의 데이터 소비자는 오리지널 변경(original change)의 결과로서 추가적 부차적 변경(side-effect change)들을 만들 뿐만 아니라 변경을 수락 또는 거절할 수 있을 것이다. 이 방식으로, 데이터는 데이터 소비자들 간에 실시간으로 동기화될 수 있을 것이다.

[0005] 데이터 피스들은 XML(Extensible Markup Language) 등과 같은 마크업 언어에 따라 구성될 수 있을 것이다. XML 스키마들은 각각의 데이터 피스와 관련이 있을 수 있고, 데이터 저장소는 정해진 데이터 피스와 관련이 있는 XML 스키마에 기반하여 데이터의 XML 구조의 유효성을 자동으로 검사할 수 있을 것이다. 이것은 무효한 변경들을 시스템에 입력하는 것이 허용되는 것을 방지함에 도움이 된다.

실 시 예

[0014] 이제 도면을 보면, 유사한 번호들이 유사한 요소들을 표시하며, 본 발명의 다양한 양태들이 기술될 것이다. 특히, 도 1 및 대응하는 설명은 본 발명의 실시예들이 구현될 수 있을 것인 적합한 컴퓨팅 환경의 간단하고 일반적인 기술을 제공하려는 것이다.

[0015] 일반적으로, 프로그램 모듈들은 특정한 작업들을 수행하거나 또는 특정한 추상 데이터 유형들을 구현하는 루틴, 프로그램, 콤포넌트, 데이터 구조, 및 다른 유형의 구조들을 포함한다. 핸드헬드 장치, 멀티프로세서 시스템, 마이크로프로세서 기반 또는 프로그래머블 가전기기, 미니컴퓨터, 메인프레임 컴퓨터 등을 포함하는 다른 컴퓨터 시스템 구성들도 이용될 수 있을 것이다. 통신 네트워크를 통해 연결된 원격 처리 장치들에 의해 작업들이 수행되는 분산 컴퓨팅 환경들이 이용될 수도 있을 것이다. 분산 컴퓨팅 환경에서, 프로그램 모듈들은 로컬 및 원격 메모리 저장 장치 모두에 배치될 수 있을 것이다.

[0016] 명세서 및 청구범위에 걸쳐, 용어의 컨텍스트가 달리 말하지 않는 한 다음의 용어들은 여기에 관련된 의미를 갖는다.

[0017] "프레젠테이션"이라는 용어는 문서가 인쇄되면 나타날 것인 텍스트 및 레이아웃 등과 같은 문서의 가시적 부분

을 지칭한다.

[0018] "태그"라는 용어는 XML 문서 내의 요소들을 개설하는 문서에 삽입된 문자들을 지칭한다. 각각의 요소는 두개의 태그, 즉 시작 태그 및 종료 태그 외에는 갖지 않을 수 있다. 하나의 태그가 허용된 경우에는 빈 요소(아무런 콘텐츠 없이)를 가질 수도 있다.

[0019] "마크업 언어" 또는 "ML"이라는 용어는 문서의 부분들이 애플리케이션에 의해 어떻게 해석될 것인지를 명기하는 문서 내의 특수한 코드들을 위한 언어를 지칭한다. 워드 프로세서 파일에서, 마크업 언어는 텍스트가 어떤 서식 또는 레이아웃을 가질 것인지를 명기한다.

[0020] "요소"라는 용어는 XML 문서의 기본 유니트를 지칭한다. 요소는 XML 문서를 위한 속성, 다른 요소, 텍스트, 및 다른 콘텐츠 영역들을 포함할 수 있을 것이다.

[0021] 태그들 사이의 XML 콘텐츠는 요소의 "자식"(또는 후손)들로 간주된다. 요소의 콘텐츠에 포함된 다른 요소들은 "자식 요소" 또는 "자식 노드" 또는 요소라고 지칭된다. 요소의 콘텐츠에 직접 포함된 텍스트는 요소의 "자식 텍스트 노드"들이라고 간주된다. 자식 요소 및 요소 내의 텍스트는 함께 그 요소의 "콘텐츠"를 이룬다.

[0022] "속성"이라는 용어는 특정한 값으로 설정되고 요소와 관련이 있는 추가적 속성을 지칭한다. 요소들은 없음을 포함하는 그들과 관련이 있는 임의의 수의 속성 설정들을 가질 수 있을 것이다. 속성들은 추가적 요소들을 포함하지 않거나 또는 텍스트 노드로서 취급될 것인 요소에 추가적 정보를 관련시키기 위해 이용된다.

[0023] "XPath"라는 용어는 XML 문서에 있는 노드들을 식별하기 위해 패턴 식을 이용하는 연산자이다. XPath 패턴은 XML 문서를 통한 경로를 기술하는 자식 요소 이름들의 슬래시로 구분된 목록이다. 패턴은 경로와 일치하는 요소들을 "선택한다".

[0024] "부차적 변경"이라는 용어는 다른 한 변경에 응답해서 만들어진 변경을 지칭한다.

[0025] "문서"라는 용어는 문서의 실제의 표면 콘텐츠를 기술하기 위해 이용될 수 있을 것인 다른 마크업 언어들뿐만 아니라 관련 있는 콘텐츠 유형에서 정해지는 속성들을 기술하는 임의의 XML로 이루어질 수 있을 것이다.

[0026] "XML 데이터 저장소 및/또는 데이터 저장소"라는 용어는, 파일이 열려 있는 동안 문서 내에 저장된 데이터의 저장 및 변경(예를 들어, XML 서식으로)을 위한 액세스를 제공하는, 워드 프로세서 문서, 스프레드시트 문서, 슬라이드 프레젠테이션 문서 등과 같은 문서 내의 컨테이너를 지칭한다. XML 데이터 저장소의 또다른 정의가 도 2와 관련하여 아래에서 제공된다.

[0027] 도 1을 보면, 본 발명을 구현하기 위한 한 예시적 시스템은 컴퓨팅 장치(100) 등과 같은 컴퓨팅 장치를 포함한다. 아주 기본적인 구성으로, 컴퓨팅 장치(100)는 통상적으로 적어도 하나의 처리 장치(102) 및 시스템 메모리(104)를 포함한다. 컴퓨팅 장치의 정확한 구성 및 유형에 따라, 시스템 메모리(104)는 휘발성(RAM 등), 비휘발성(ROM, 플래시 메모리 등) 또는 둘의 어떤 조합일 수 있을 것이다. 시스템 메모리(104)는 통상적으로 운영 체제(105), 하나 이상의 애플리케이션(106)을 포함하고, 프로그램 데이터(107)를 포함할 수 있을 것이다. 한 실시예에서는, 애플리케이션(106)이 워드 프로세서 애플리케이션(120)을 포함할 수 있을 것이다. 이 기본적인 구성은 도 1에서 점선(dashed line)(108) 내의 콤포넌트들에 의해 예시되어 있다.

[0028] 컴퓨팅 장치(100)는 또다른 특징 또는 기능을 가질 수 있을 것이다. 예를 들어, 컴퓨팅 장치(100)는 예를 들어, 자기 디스크, 광디스크, 또는 테이프 등과 같은 추가적 데이터 저장 장치(이동식 및/또는 비이동식)를 포함할 수도 있을 것이다. 그러한 추가적 저장장치는 도 1에서 이동식 저장장치(109) 및 비이동식 저장장치(110)에 의해 예시되어 있다. 컴퓨터 저장 매체는 컴퓨터 판독 가능 명령어, 데이터 구조, 프로그램 모듈, 또는 다른 데이터 등과 같은 정보의 저장을 위한 어떤 방법 또는 기술로든 구현되는 휘발성 및 비휘발성, 이동식 및 비이동식 매체를 포함할 수 있을 것이다. 시스템 메모리(104), 이동식 저장장치(109) 및 비이동식 저장장치(110)는 모두 컴퓨터 저장 매체의 예이다. 컴퓨터 저장 매체는, 원하는 정보를 저장하기 위해 이용될 수 있고 컴퓨팅 장치(100)에 의해 액세스될 수 있는, RAM, ROM, EEPROM, 플래시 메모리 또는 다른 메모리 기술, CD-ROM, DVD(digital versatile disk) 또는 다른 광 저장장치, 자기 카세트, 자기 테이프, 자기 디스크 저장장치 또는 다른 자기 저장 장치, 또는 다른 어떤 매체든 포함하지만, 제한적인 것은 아니다. 그러한 어떤 컴퓨터 저장 매체든 장치(100)의 부분이 될 수 있을 것이다. 컴퓨팅 장치(100)는 또한 키보드, 마우스, 펜, 음성 입력 장치, 터치 입력 장치 등과 같은 입력 장치(112)를 가질 수도 있을 것이다. 디스플레이, 스피커, 프린터 등과 같은 출력 장치(114)도 포함될 수 있을 것이다. 이러한 장치들은 당 기술 분야에서 잘 알려진 것이고 여기에서 길게 설명할 필요가 없다.

[0029] 컴퓨팅 장치(100)는 장치가 네트워크를 통하는 것 등과 같이 다른 컴퓨팅 장치(118)들과 통신하게 하는 통신 접속부(116)를 포함할 수도 있을 것이다. 통신 접속부(116)는 통신 매체의 한 예이다. 통신 매체는 통상적으로 컴퓨터 판독 가능 명령어, 데이터 구조, 프로그램 모듈, 또는 반송파 또는 다른 전송 메커니즘 등과 같은 변조된 데이터 신호에 있는 기타의 데이터에 의해 구현될 수 있을 것이며, 어떤 정보 전달 매체든 포함한다. "변조된 데이터 신호"라는 용어는 그 특성들 중 하나 이상을 정보를 신호로 인코딩하는 방식으로 설정 또는 변경한 신호를 의미한다. 예를 들어, 통신 매체는 유선 네트워크 또는 직접 배선 접속 등과 같은 유선 매체 및 음향, RF, 적외선 및 기타의 무선 매체 등과 같은 매체를 포함하지만, 제한적인 것은 아니다. 여기에서 이용되는 컴퓨터 판독 가능 매체라는 용어는 저장 매체 및 통신 매체를 모두 포함한다.

[0030] Washington, Redmond에 소재하는 MICROSOFT 사로부터의 WINDOWS 운영 체제 등과 같은 네트워크 된 퍼스널 컴퓨터의 운영을 컨트롤하기에 적합한 운영 체제(105)를 포함하는 다수의 프로그램 모듈 및 데이터 파일들이 컴퓨팅 장치(100)의 저장된 시스템 메모리(104)에 저장될 수 있을 것이다. 시스템 메모리(104)는 워드 프로세서 애플리케이션(120) 및 아래에서 기술된 기타의 것 등과 같은 하나 이상의 프로그램 모듈들도 저장할 수 있을 것이다. 워드 프로세서 애플리케이션(120)은 전자 문서들을 만들고 편집하며 처리하기 위한 기능을 제공하도록 작동한다.

[0031] 본 발명의 한 실시예에 따르면, 워드 프로세서 애플리케이션(120)은 MICROSOFT 사로부터의 WORD 프로그램을 포함한다. 그러나, 다른 제조업자들로부터의 워드 프로세서 애플리케이션 프로그램들이 활용될 수도 있을 것임을 알아야 한다. 워드 프로세싱 애플리케이션의 예시는 단지 예시의 목적일 뿐이며, 문서들을 생성하거나 문서 상에서 조작할 수 있는 다른 유형의 애플리케이션들을 제한하는 것이 아니다. 예를 들어, 스프레드시트 애플리케이션 프로그램, 데이터베이스 애플리케이션 프로그램, 슬라이드 프레젠테이션 애플리케이션 프로그램, 도면 또는 컴퓨터 보조 애플리케이션 프로그램 등과 같은, 다양한 형태의 콘텐츠(예를 들어 텍스트, 이미지, 사진 등)들을 처리할 수 있는 다른 애플리케이션 프로그램(106)이 동등하게 적용될 수 있다. 다양한 상이한 유형의 문서들을 생성 및 조작하는 예시적 애플리케이션 프로그램(106)은 MICROSOFT 사로부터의 OFFICE를 포함한다.

[0032] 실시예들은 컴퓨터 프로세스, 컴퓨팅 시스템, 또는 컴퓨터 프로그램 제품 또는 컴퓨터 판독 가능 매체 등과 같은 제품으로서 구현될 수 있을 것이다. 컴퓨터 프로그램 제품은 컴퓨터 시스템에 의해 판독할 수 있고 컴퓨터 프로세스를 실행하기 위한 명령어의 컴퓨터 프로그램을 인코딩한 컴퓨터 저장 매체일 수 있을 것이다. 컴퓨터 프로그램 제품은 컴퓨팅 시스템에 의해 판독할 수 있고 컴퓨터 프로세스를 실행하기 위한 명령어의 컴퓨터 프로그램을 인코딩한 반송파 상에서 전파되는 신호일 수도 있을 것이다.

[0033] 도 2는 하나 이상의 클라이언트 애플리케이션과 하나 이상의 데이터 저장소 및 데이터 저장소의 콘텐츠들 사이의 관계를 예시하는 블록 다이어그램이다. 일반적으로 말해서, 데이터 소비자들 사이에서 컴퓨터 생성 문서와 관련이 있는 임의의 데이터를 저장하고 관계를 맺으며 이용하게 하기 위해 문서 내의 주 프레젠테이션 저장장치와 별도로 하나 이상의 데이터 저장소들이 유지된다. 문서 메타데이터 등과 같은, 문서와 관련이 있는 정보를 구성하기 위한 데이터가 상이한 데이터 피스들 사이의 관계가 유지된 데이터 저장소에 유지된다. 데이터 저장소는 상이한 애플리케이션들이 하나 이상의 데이터 피스에 액세스 및 조작하게 하기 위해 데이터 저장소의 다양한 데이터 피스들에게 API(application programming interface)들을 노출시킨다. 여기에서 이용되는 "데이터 소비자", "애플리케이션" 및 "프로세스"라는 용어는 컨텍스트가 명백하게 달리 말하지 않는 한 호환적으로 이용될 수 있을 것이다.

[0034] 데이터 피스들은 XML(Extensible Markup Language) 등과 같은 마크업 언어에 따라 구성될 수 있을 것이다. XML 스키마들은 각각의 데이터 피스와 관련이 있을 수 있고, 데이터 저장소는 각각의 요청의 유효성을 보장하기 위해 정해진 데이터 피스와 관련이 있는 XML 스키마에 기반하여 데이터에 적용된 XML 구조의 유효성을 검사할 수 있을 것이다. 데이터 저장소들은 예를 들어 XML(Extensible Markup Language)에 따라 구성된 메타데이터와 같은 어떤 수의 임의의 데이터 아이템들이든 포함할 수 있을 것이다. 따라서, 문서 솔루션 제공자들은 임의의 메타데이터를 정해진 문서와 함께 XML로서 저장하고, 데이터가 데이터 저장소에 로드되거나 제거될 때 및/또는 문서가 사용자에 의해 열리거나 편집되거나 저장될 때 등과 같은 이벤트의 발생시 데이터에 대한 액세스를 갖는 정해진 솔루션에 의해 그 정보를 처리할 수 있을 것이다.

[0035] 문서가 편집되는 동안에 그 XML 형태의 데이터에 대한 프로그러매틱 액세스(programmatic access)가 제공되기도 한다. 한 실시예에 따르면, 문서가 열려 있는 동안 데이터에 액세스하여 프로그러매틱 변경을 하기 위해 이용할 수 있는, 솔루션 개발자들에게 친숙한 표준 메커니즘이 제공된다. 이 프로그러매틱 액세스는 모방적(mimic) 표준 XML API들로 디자인된다. 데이터에 대한 프로그러매틱 액세스는 하나 이상의 편집용 클라이언트 애플리케

이션(예를 들어, 문서 편집 또는 만들기 애플리케이션 및/또는 제3자 애플리케이션 애드인 솔루션 등)들에 대한 API들에 의해 제공된다. 따라서, 다수의 클라이언트 애플리케이션들이 동일한 문서 데이터 피스에 액세스하여 편집할 수 있을 것이고, 정해진 데이터 피스에 대한 어떤 저촉되는 변경들이든 해결된다. 데이터 소비자들은 어떤 정해진 변경에든 응답하여 부차적 변경들을 만들 수 있을 것이다. 예를 들어, 회사명을 "MICROSOFT"로 설정하는 것에 응답하여, 데이터 소비자는 주식 기호(stock symbol)을 "MSFT"로 변경할 수 있을 것이다. 또한, 데이터에 대한 변경 및 어떤 관련 있는 부차 효과(side-effect)들이든 하나 이상의 변경들을 실행취소(undo)하는 것이 관계 있는 모든 변경을 번복하도록 데이터 저장소에 의해 "번들(bundle)"로 될 수 있을 것이다. 이것은 사용자가 예를 들어, 실행취소 명령(Undo command)을 누름으로써 문서 표면(document surface)으로부터 오리지널 변경의 실행취소를 시작할 때 모든 변경들을 번복했음을 보장하기 위한 데이터 소비자 자신으로부터의 전개(development)의 부담을 제거하는 것에 도움이 된다.

[0036] 문서 데이터에 적용된 XML 데이터가 유효함을 보장하기 위해 문서 메타데이터와 관련이 있는 어떤 사용자 지정 XML 데이터 피스의 콘텐츠들이든 정의하기 위한 표준 XML 스키마(XSD)들이 이용될 수도 있을 것이다. 이러한 스키마들은 문서에 저장된 어떤 XML 데이터의 인스턴스에든 첨부될 수 있을 것이며, 데이터 저장소는 그 데이터의 XML 구조로 귀결될 XML 데이터에 대한 어떤 변경이든(즉, 그들의 콘텐츠에 대립되는 XML 태그) 무효하게 되는 것을 불허하도록 구성될 수 있을 것이다. 이것은 솔루션 개발자가 문서에 대해 특수한 XML 메타데이터 피스를 부착할 수 있는 것을 보장하고, 그 데이터를 변경하기 위해 데이터 소비자(예를 들어, 애드인)들이 이용되는지에 무관하게 관련 스키마에 따라 계속 구조적으로 "수정"될 것임을 보장하는 것을 돕는다. 스키마는 파일 속 또는 하드 드라이브 등과 같은 컴퓨터 판독 가능 매체에 저장될 수 있을 것이다.

[0037] 이제 도 2를 보면, 문서 데이터(220)가 XML 구조 데이터 및 관련된 문서의 표면 또는 프레젠테이션 레벨 뷰(surface or presentation level view)를 표시하는 문서 데이터를 포함한다. 예를 들어 문서 데이터(220)는 워드 프로세싱 문서, 스프레드시트 문서, 슬라이드 프레젠테이션 문서 등의 XML 구조(예를 들어, 헤딩 태그, 본문 태그, 종결 태그) 및 관련된 표면 뷰 데이터(예를 들어, 단어, 문장, 단락)를 포함할 수 있을 것이다.

[0038] 데이터 저장소(208)는 정해진 문서와 관련이 있는 하나 이상의 데이터 유형과 관련이 있는 하나 이상의 구조적 데이터 피스를 저장하기 위한 문서 데이터 리포지토리(document data repository)이다. 단 하나의 데이터 저장소가 예시되어 있을지라도, 하나를 초과하는 데이터 저장소가 활용될 수 있을 것이다. 메타데이터 1(225)(구조적 데이터 아이템)은 문서와 관련이 있는 제1 메타데이터 피스를 위한 XML 구조 데이터 및 관련 데이터를 포함할 수 있을 것이다. 예를 들어, 메타데이터 1(225)은 문서 만든 이(document author), 문서 생성 일자(date of document creation), 문서 최종 변경/저장 일자(date of document last change/save) 등을 나열하는 XML 구조 데이터(예를 들어, 일자 태그, 이름 태그 등)를 포함할 수 있을 것이다. 메타데이터 2(230)(구조적 데이터 아이템)는 문서와 관련이 있는 제2 메타데이터 피스를 표시하는 XML 구조 데이터(태그) 및 관련 메타데이터를 포함할 수 있을 것이다. 메타데이터 1 및 메타데이터 2는 예시를 위한 것이며 정해진 문서에 관한 데이터 저장소(208)에 유지될 수 있는 상이한 데이터 유형의 다양성 및 수에 대한 제한이 아니다. 예를 들어, 여기에 기술된 바와 같이, 문서 데이터에 대한 액세스를 갖는 솔루션 제공자 또는 사용자들이 원하는 바와 같이 하나 이상의 소프트웨어 애플리케이션에 의해 임의의 데이터가 구성되어 문서에 추가될 수 있을 것이다.

[0039] 스키마 파일(schema file)(240, 245)은 각각의 데이터 피스(225, 230)에 적용된 XML(Extensible Markup Language) 데이터와 관련이 있는 신택스 및 유효성 검사 규칙(syntax and validation rule)들을 지시하기 위해 데이터 저장소(208)에 저장된 각각의 데이터 피스에 선택적으로 부착될 수 있을 것이다. XML 스키마 파일들은 XML 환경에서 데이터를 기술하고 유효성을 검사하기 위한 방식을 제공한다. 스키마 파일은, 요소 및 속성을 포함해서, 어떤 XML 마크업 데이터가 XML 문서의 콘텐츠를 기술하기 위해 이용되는지를 정하고, 스키마 파일은, 언제 각각의 요소가 허용되는지, 어떤 콘텐츠 유형이 요소 내에서 허용되는지, 및 어떤 요소들이 다른 요소 내에 나타날 수 있는지를 포함해서, XML 마크업 신택스를 정의한다. 스키마 파일들의 이용은 문서(또는 개별적 데이터 피스)가 일관되고 예측 가능한 방식으로 구성됨을 보장한다. 스키마 파일(240, 245)들은 사용자에 의해 만들어질 수 있을 것이며, 일반적으로 XML 등과 같은 관련 마크업 언어에 의해 지원된다.

[0040] 문서의 이러한 체계화(schematization)는 데이터 저장소가 데이터 저장소 레벨에서 정해진 스키마 파일을 어기는 어떤 변경이든 거절함으로써 문서의 구조적 유효성을 "보장"할 능력을 제공하게 한다. 한 실시예에 따르면, 데이터 저장소(208)가, 관련 스키마 파일을 거슬러 정해진 데이터 피스에 대해 이루어진 변경 또는 추가된 XML 구조의 유효성을 검사하기 위해, 스키마 유효성 검사 모듈(schema validation module)(260)을 활용한다. 예를 들어, 문서 작성자 또는 편집자가 정해진 데이터 피스, 예를 들어 메타데이터 1에 대한 XML 구조적 변경들을 만들면 - 편집자가 정해진 XML 태그를 추가 또는 제거함 -, 데이터 저장소(208)는 변경의 유효성을 보장하도

록 관련 스키마 파일을 거스르는 XML 구조적 변경들을 검사하기 위해 스키마 유효성 검사 모듈을 활용할 것이다. 변경이 유효하지 않으면, 편집자에게 오류가 발생될 수 있다. 알고 있듯이, 정해진 데이터 피스에 적용되는 XML 구조의 그러한 컨트롤은 클라이언트 및 제3자 애플리케이션들이 관련 데이터와 상호작용하게 하기 위해 특히 중요한 구조적 일관성 및 예측 가능성을 허용한다. 어떤 데이터 소비자든 데이터의 유효성을 검사하기 위해 이용될 수 있는 스키마를 제공할 수 있을 것이다.

[0041] 데이터 저장소(208)는 개개의 애플리케이션(205, 210, 215)들의 OM(object model)들에 의해 제3자 애플리케이션(210, 215)뿐만 아니라 클라이언트 애플리케이션(205)(예를 들어, 워드 프로세싱 애플리케이션, 스프레드시트 애플리케이션, 슬라이드 프레젠테이션 애플리케이션 등)들에 의해 액세스될 수 있는 하나 이상의 API(application programming interface)(270)들을 제공한다. 이러한 API들은 클라이언트 애플리케이션 및 제3자 애플리케이션들이 어떤 기존의 XML 파일이든 정해진 문서의 데이터 저장소(208)에 로드하게 하고, 그래서 그 데이터가 이제 문서의 일부이며 그 존속기간(예를 들어, 열기/편집/저장/이름 바꾸기 등) 동안 또는 데이터가 데이터 저장소로부터 삭제되기까지 그 문서 내에서 전해질 것임을 보장한다. 한 실시예에 따르면, 데이터 저장소 내의 데이터는, 정해진 데이터 피스(225, 230)를 위한 소스 애플리케이션이 닫히거나 또는 다른 방식으로 이용 불가할 때조차도, 그 XML 서식으로 이용 가능하다. 즉, 정해진 데이터 피스(225, 230)는 한 세트의 API들에 의해 액세스될 수 있을 것이다. 아래에 기술된 바와 같이, API들은 또한 클라이언트 및 제3자 애플리케이션들이 데이터 아이템(225, 230)들에 적용된 XML 마크업 데이터에 대한 변경을 만들게 한다.

[0042] XML 데이터(225, 230)가 문서(220)와 관련해서 데이터 저장소에 로드되면, 그것은 개발자들의 XML 프로그래밍 표준에 관한 기존의 지식을 활용하기 위해 기존의 XML 편집용 인터페이스들과 유사한 방법을 제공하도록 디자인된 데이터 저장소 인터페이스를 이용하여 표준 XML로서 조작될 수 있다. 이것은 사용자들이, 요소 및 속성들을 추가하고, 요소 및 속성들을 제거하며, 기존의 요소/속성들의 값을 변화시키고, 관련 XML 트리의 어떤 기존의 부분의 값들이든 판독하는 것 등과 같이, 문서를 위해 데이터 저장소에 추가된 XML 데이터에 대한 표준 XML 조작을 수행하게 한다. 이러한 XML 표준 조작들을 이용하여, 솔루션들은 구조적 복합 메타데이터를 문서와 함께 저장할 수 있을 것이다. 예를 들어, 각각의 문서를 위해 데이터 저장소(208)에 추가된 메타데이터 1(225)을 판독함으로써 다수의 문서들로부터 문서 만든 이 이름 및 문서 생성 일자들을 찾고 추출하기 위해 제3자 애플리케이션(215)이 쓰여질 수 있을 것이다. 예시적 제3자 애플리케이션은 정해진 조직에 의해 만들어진 모든 문서 들을 위한 문서 만든 이 이름 및 문서 생성 일자들의 목록을 만들기 위해 프로그램된 애플리케이션일 수 있을 것이다. 본 발명의 실시예들에 따르면, 제3자 애플리케이션은 원하는 데이터를 효율적으로 찾고 추출하기 위해 메타데이터 1에 적용된 XML 구조를 활용할 수 있을 것이다. 예를 들어, 그러한 태그들과 관련이 있는 데이터를 얻고 이용하기 위한 <docauthor> 및 <doccreationdate> 등과 같은 XML 태그들을 찾기 위해 메타데이터 1 파일의 XML 구조를 분석하기 위해 제3자 애플리케이션이 쓰여질 수 있을 것이다. 알 수 있듯이, 앞서의 것은 하나 이상의 애플리케이션들이 데이터 저장소(208)에 의해 문서와 관련이 있는 구조적 데이터와 상호작용할 수 있는 많은 방식들 중 단지 한 예이다.

[0043] 또한, 다수의 애플리케이션(205, 210, 215)들이 동일한 데이터 피스와 함께 작업할 수 있도록 데이터 저장소(208)는 XML 데이터(220, 225, 230)의 어떤 개별적 피스(저장소 아이템이라고도 알려짐)에 대해서든 어떤 수의 API 인터페이스(270)들이든 제공한다. 예를 들어, 클라이언트 애플리케이션(예를 들어, 워드 프로세싱 애플리케이션) 및 제3자 애플리케이션 솔루션(예를 들어, 앞서 기술된 제3자 애플리케이션) 등과 같은 몇몇 솔루션들은 동일한 세트의 문서 속성(예를 들어, 메타데이터 2(230) 파일에 포함된 속성)들과 함께 작업할 수 있을 것이다. 데이터 저장소(208)를 이용하면, 이러한 애플리케이션들의 각각은, 각각의 애플리케이션이 동일한 데이터 피스에 액세스하는 다수의 데이터 소비자들을 가져야 하는 복잡함을 처리할 필요 없이 자신의 개체 관리자에 의해 데이터와 통신하게 하기 위해 그들 자신의 데이터 저장소 API 인터페이스(270)를 통해 원하는 XML 데이터(230)에 대한 개별적인 액세스를 수신한다.

[0044] 이러한 다수의 애플리케이션(205, 210, 215)들이 동일한 데이터에 액세스하게 하기 위해, 데이터 저장소(208)는 XML 데이터의 어떤 부분이든 다른 한 애플리케이션에 의해 변경될 때 이러한 애플리케이션들의 각각에게 알려주어 정해진 애플리케이션이 그 변경에 응답(자신의 프로세스에 대해 내부적으로 및 동일한 데이터에 대한 다른 변경에 의해 외부적으로)할 수 있게 한다. 한 애플리케이션이 정해진 데이터 아이템에 대한 변경을 요청할 때, 그 요청은 다른 모든 애플리케이션들에게 자동으로 보내져 다른 애플리케이션들이 요청된 변경에 응답할 것인지 또는 어떻게 응답할 것인지를 결정하게 한다. 한 실시예에 따르면, 이것은 각각의 애플리케이션이 자신이 인터페이스를 갖는 XML 데이터의 어떤 부분이든 “듣기” 위해 등록하게 하여 정해진 애플리케이션 솔루션/프로그램만 자신의 논리(logic)에 적합한 메시지들을 수신하도록 함으로써 이루어진다. 예를 들어, 한 유형의 애플리케

이션(210)은 상세한 비니지스 논리 능력들을 제3자 솔루션에 제공하기 위해 정해진 XML 데이터에 대해 만들어진 모든 변경들을 듣기 위해 등록하고자 할 수 있을 것이지만, 다른 한 유형의 애플리케이션(215)은 자신의 논리가 XML 데이터의 다른 어떤 부분에 대한 변경에 관해서든 무관하기 때문에 동일한 데이터 내의 하나 또는 두개의 특수한 XML 요소들에 대한 변경만을 듣고자 할 수 있을 것이다.

[0045] 이 실시예에 따르면, 다수의 애플리케이션(205, 210, 215)들이 동일한 문서 데이터 피스에 액세스하여 편집할 수 있고, 정해진 데이터 피스에 대한 어떤 저촉적인 변경들이든 해결된다. 예를 들어, 한 애플리케이션에 의한 한 변경이 다른 한 애플리케이션에 의한 부차적 변경을 유발할 때 어떤 정해진 변경에 대한 "부차 효과"든 만들어질 수 있을 것이다. 예를 들어, 제1 애플리케이션(210)은, 정해진 문서에 관한 회사 주식 기호들의 목록을 컴파일하기 위해, 가능하다면, 회사 이름들을 대응하는 주식 기호들로 변환하기 위해 정해진 문서와 관련이 있는 하나 이상의 데이터 아이템(225, 230)들로부터 회사 이름들을 추출하는 작업이 주어질 수 있을 것이다. 제2 애플리케이션(215)이 예를 들어, "Company ABC"에서 “Company XYZ"로 회사 이름을 변경하여 정해진 메타데이터 피스에 있는 정해진 회사 이름이 추가 또는 변경되게 한다면, 제1 애플리케이션은 그 주식 기호들의 목록이 " Company ABC" 대신에 " Company XYZ"를 위한 주식 기호를 포함하도록 자동으로 업데이트하기 위해 이 변경을 들을 수 있을 것이다. 또한, 그러한 변경 및 어떤 관련 있는 부차 효과들이든 데이터 저장소(208)에 의해 번들화하여 하나 이상의 변경들이 모든 관련 있는 변경들을 번복시키게 할 수 있을 것이다.

[0046] 도 3은 XML 데이터 저장소들을 갖는 내부 및 외부의 데이터 소비자들 사이의 상호작용을 도시하는 시스템 다이어그램을 예시한다. 예시된 바와 같이, 시스템(300)은 문서(315), 데이터 저장소(302), 프레젠테이션 계층(304), 데이터 저장소(202) 내의 XML 저장소 1-N(306) - 각각 오류 저장소(error store) 및 실행취소 저장소(undo store)를 포함함 -, 글로벌 변경 저장소(global change store)(308), 선택적 글로벌 실행취소 저장소(optional global undo)(310), 내부의 데이터 소비자 1-N(314)에 연결된 내부의 브로커(internal broker)(312), XML 외부의 저장소(320) 및 외부의 데이터 소비자 1-N(318)에 연결된 외부의 브로커(external broker)(316)를 포함한다.

[0047] 데이터 저장소(302) 및 데이터 저장소(306)를 이용하여, 문서들은 어떤 수의 임의의 데이터 아이템(각각 표준 XML 신택스에 일치하는 한)들이든 포함할 능력을 갖는다. 임의의 메타데이터는 문서 내에 XML로서 저장될 수 있고 그 정보는 문서가 사용자에 의해 열기/편집/저장될 때 자동으로 라운드 트립(round-trip)될 수 있을 것이다.

[0048] 앞서 설명했듯이, 문서가 열려 있는 동안에 정보에 프로그러매틱하게 액세스하여 변경하기 위해 이용할 수 있는 솔루션 개발자들에게 친숙한 표준 메커니즘을 제공하여, 문서가 편집되는 동안에 활용될 수 있을 것인 API에 의해 이 데이터에 대한 프로그러매틱 액세스가 제공된다. 한 실시예에 따르면, 이 프로그러매틱 액세스는 모방적인 표준 XML 인터페이스로 디자인된다. API를 이용하여, 워드 프로세싱 애플리케이션 등과 같은 애플리케이션이 실행중일 때 데이터가 추가/제거될 수 있고, 저장소 아이템(데이터 저장소 내의 일부) 내에 데이터가 거주될 수 있으며, 표준 XML 구성들을 이용하여 데이터가 조작될 수 있고, 스키마들은 데이터 저장소에 있는 어떤 임의의 XML 데이터에든 관련이 있을 수 있으며, 데이터 저장소 아이템과 관련이 있으면 스키마들이 추가/제거될 수 있고, XML 변경들은 듣고 있는 어떤 클라이언트들에 대해서든 이벤트 될 수 있다. 예시된 바와 같이, API는 데이터 저장소(302)와 상호작용하는 외부의 데이터 소비자(318)들을 위한 인터페이스를 제공하는 외부의 브로커(316) 및 내부의 데이터 소비자(314)들을 위한 인터페이스를 제공하는 내부의 브로커(312)를 포함한다.

[0049] 데이터 저장소(302)에 대한 조작은 실시간으로 일어날 수 있을 것이다. 앞서 설명한 바와 같이, 데이터 저장소(302 및 306)들은 하나 이상의 유형의 데이터를 포함할 수 있을 것이다. 예를 들어, 한 회사는 단일의 데이터 저장소 내에 저장하고자 하는 상이한 유형의 데이터를 모두 저장하기 위해 이용하고 있는 하나의 데이터 저장소를 가질 수 있을 것임에 반해, 다른 한 회사는 상이한 유형들의 데이터를 상이한 데이터 저장소들에 저장하고자 할 수도 있을 것이다.

[0050] 데이터 소비자(내부(314) 및/또는 외부(318))들은 데이터 저장소 내의 데이터 관련 동작들에 관한 이벤트들을 위해 등록할 수 있을 것이다. 예를 들어, 데이터 소비자는 하나 이상의 데이터 저장소에 대해 어떤 유형의 변경이든 이루어질 때 이벤트를 수신하기 위해 등록할 수 있을 것이다. 다른 한 데이터 소비자는 데이터 저장소 내의 특정한 요소 또는 요소 세트에 대해 일어나는 변경들을 위해 등록할 수 있을 것이다. 보편적인 이벤트는 데이터 저장소들 중 하나에 대한 아이템 추가, 아이템 변경, 및 아이템 삭제를 포함한다. 이벤트가 일어날 때, 등록한 각각의 데이터 소비자는 변경에 반응할 수 있을 것이지만, 데이터 저장소들의 상태는 일관되게 유지된다. 종종, 데이터 소비자는 변경이 이루어질 때 어떤 동작도 수행하지 않을 것이다. 다른 경우에는, 데

이터 소비자가 이벤트에 응답하여 어떤 동작들을 수행할 것이다. 예를 들어, 데이터 소비자는 명칭 변경에 응답하여 문서 내의 헤더를 업데이트하는 것 등과 같이 변경에 응답하여 데이터 저장소에 대한 어떤 다른 변경들을 만들 수 있을 것이다. 데이터 소비자는 문서에 영향을 주지 않는 어떤 다른 조작을 수행할 수도 있을 것이다. 예를 들어, 주식 티커 기호(stock ticker symbol)가 삽입되면, 데이터 소비자는 모든 검색된 데이터가 프레젠테이션 계층에서의 문서 내에 디스플레이되지는 않을지라도 주식 기호와 관련이 있는 데이터를 검색할 수 있을 것이다. 데이터 소비자는 그 자신의 유효성 검사 논리를 이용하여 변경을 거절할 수도 있을 것이다. 예를 들어, 데이터 소비자 1이 그가 수락하지 않은 변경을 수신하면, 그 데이터 소비자는 변경이 수락되지 않았음을 나타내는 플래그(flag)를 브로커에게 반환할 수 있을 것이다. 변경이 수락되지 않을 때마다, 어떤 부차 효과와 함께든 변경이 롤백(roll back)되어, 변경이 일어나지 않게 한다. 각각의 XML 저장소(306)는 그것이 만든 변경들을 실행취소하기 위해 그 실행취소 저장소를 활용할 수 있을 것이다. 대안적으로, 여러 데이터 저장소들에 있는 변경들을 실행취소하기 위해 글로벌 실행취소 저장소(310)가 활용될 수 있을 것이다. 문서 속성들에 대해 무엇이 일어났는지에 관심이 있는 3개의 데이터 소비자들이 있다고 상상하자. 변경이 만들어지면, 데이터 저장소는 각각의 데이터 소비자가 등록했는지를 결정하고 그들의 각각에게 예정된 순서대로 알린다. 이어서, 각각의 데이터 소비자는 변경에 응답하여 어떤 동작을 수행할 수 있을 것이다. 변경, 또는 변경의 결과로서 등록된 데이터 소비자들에 의해 만들어지는 어떤 변경이든 데이터 소비자들 중 어느 누구에 의해서도 수락되지 않으면, 초기의 변경에 관한 모든 변경들이 실행취소된다.

[0051] 외부의 브로커 API(application programming interface) 계층(316)은 외부의 데이터 소비자(318)에 의한 데이터 저장소(302)에 대한 액세스를 제공하고 제3자 클라이언트들이 애플리케이션과 관련이 있는 내부의 데이터 소비자들이 데이터 저장소와 상호작용하는 것처럼 데이터 저장소(302)와 상호작용하게 한다. 데이터 저장소(302) 내의 각각의 XML 데이터 저장소(306)에게는 식별을 위한 고유의 ID가 제공된다. 이것은 XML 데이터 저장소(306)를 찾는 것을 돕는다.

[0052] 어떤 시점에서든, 데이터 소비자는 데이터 저장소 내의 데이터의 유효성을 검사하기 위해 이용되는 스키마를 추가할 수 있을 것이다. 스키마가 추가되고 데이터 소비자가 어떤 데이터에 대해서든 변경하려고 시도할 때, 데이터 저장소는 변경이 제공된 스키마와 뜻이 맞는지를 결정한다. 스키마가 부착되면, 브로커는 유효성 검사 개체(validating object)로 된다.

[0053] 사용자 정의 XML 스키마들은 워드 프로세싱 문서, 스프레드시트 문서 등과 같은 문서 내의 콘텐츠들의 주위에 시맨틱 마크업을 제공하기 위해 이용될 수 있을 것이다. 이 강력한 기능은 개발자들이 그들의 솔루션이 기본적인 애플리케이션의 프레젠테이션 형식의 복합성을 취급할 것을 요구하는 대신에 그들의 데이터의 구조 및 콘텐츠에 대해 직접 작업하기 위해 이 사용자 지정 XML 포함(embedding)을 활용하는 솔루션들을 만들게 한다.

[0054] 예를 들어, 사용자가 XML 능력이 없는 애플리케이션에서 지분 조사서를 위한 커버 페이지를 만들고자 했다면 유용한 데이터(예를 들어, 회사 이름, 주식 티커 기호, 주식 등급(stock rating))를 추출하는 것은 문서의 프레젠테이션 형식에 밀접하게 관련된 애플리케이션의 개체 모델의 이용을 요구했을 것이다. 이것은 도출되는 솔루션 논리도 문서의 프레젠테이션 형식에 관련이 있고, 그 프레젠테이션이 변경되어야 한다면 실패함을 의미했다. 예를 들어, 코드가 스티커 기호가 제1 표의 행 3, 열 2에 있을 것으로 기대한다면, 새로운 열/행을 추가하는 것은 이 논리를 어길 것이다. 그러나, XML 능력이 있는 애플리케이션들의 경우에, 이 코드는 논리가 프레젠테이션에 관련될 필요성을 없애고 고객 자신의 데이터의 구조에 링크될 수 있다. 그 동일한 논리가 <stockSymbol/> XML 노드의 콘텐츠를 위해 조사할 수 있고, 그 컨텍스트 프레젠테이션이 철저히 변경되었을지라도 그것이 문서의 어디에 있든지 그것을 편집하기 위해 찾을 수 있다.

[0055] XML 스키마는, 메타데이터(예를 들어, 저장/처리하기 위한 만든 이 데이터), 본문 데이터(예를 들어, 보고되는 회사) 및 표 데이터(예를 들어, 주가 이력)을 포함해서, 흔히 몇몇 유형의 데이터를 포함한다. 그러나, 이러한 데이터 유형들은 서로 배타적인 것이 아니다. 사실상, 그들은 통상적으로 동일한 XML 문서 내의 영역들을 넓게 오버랩하고 있다. 원칙적으로, 이 데이터가 단일의 XML 스키마에 의해 모두 표현될지라도, 이러한 다양한 데이터 '유형'들의 각각은 그 데이터의 가장 바람직한 식(optimal expression)으로 확실하게 맞춰진 환경에서 편집될 수 있다. 예를 들어, 한 형식이 사용자가 문서를 위한 메타데이터를 용이하게 편집하도록 허용하게 보일 수 있는 반면에, 문서 본문은 워드 프로세싱 애플리케이션에 의해 편집할 수 있다. 이것은 사용자가 문서 및 형식의 부분들을 동시에 채울 수 있도록 실시간으로 일어난다.

[0056] 데이터 저장소들은 하나를 초과하는 요소를 한번에 수신할 수도 있을 것이다. 특정한 스트림으로서의 데이터(XML)를 제공하는 것은 어떤 상황에서 스키마를 충족시키는 것을 도울 수 있게 한다. 예를 들어, 부착된 스키

마가 주식 데이터가 존재한다면 그것이 적어도 두개의 회사들을 가져야 함을 말한다고 가정하자. 주식 데이터가 하나씩 추가되었다면 그것은 유효하지 않을 것이다.

[0057] 한 실시예에 따르면, 데이터의 유효성을 검사하기 위해 단일의 경로가 이용된다. 데이터 저장소에 대한 변경이 이루어지는 변경으로 귀결될 수 있는 두개의 경로를 만드는 대신에, 데이터가 데이터 저장소에 커밋되기 전에 유효성 검사가 수행된다. 이것은 데이터 소비자가 데이터 저장소에 오류를 도입하는 것을 방지하는 것을 돕는다.

[0058] 앞서 설명했듯이, 문서와 관련이 있는 XML 데이터가 중앙의 XML 데이터 저장소(302)에 있는 어떤 특수한 애플리케이션 문서로부터든 따로 저장될 수 있다. 하나의 XML 데이터 피스의 프레젠테이션/편집을 위해 다수의 환경이 만들어질 수 있을 것이다. 그 데이터의 식(expression)들은 XML 데이터 저장소에 있는 동일한 데이터에 대한 그들의 접속을 통해 자동으로 동기화된다. 그와 같이, 다수의 애플리케이션들이 동일한 기본적인 XML 데이터를 동시에 디스플레이할 수 있다. 이것은 사용자에게 애플리케이션에 있는 동일한 데이터를 편집하기 위한 능력, 즉 '그 일을 위한 최선의 도구'가 제공됨을 의미한다. 예를 들어, 메타데이터 정보를 편집하기 위한 형식, 데이터의 자유형식 섹션들을 편집하기 위한 워드 프로세싱 문서 표면 등. 이것은 또한 사용자가 원하는 다수의 애플리케이션에 있는 데이터를 편집할 수 있음을 의미한다. 동일한 정보가 다수의 애플리케이션에 나타나면, 사용자는 그들의 현재의 편집 컨텍스트에 기반하여 원하는 바에 따라 그들 중 어느 것에 있든 그 데이터를 편집할 수 있다.

[0059] 각각의 애플리케이션이 문서와 관련이 있는 모든 XML 데이터에 대한 동시적 액세스를 가질지라도, 각각의 애플리케이션은 그 데이터의 어떤 부분을 디스플레이하고 편집할 것인지의 선택을 개별적으로 만들 수 있다. 이것은 각각의 애플리케이션이 그 컨텍스트 내의 관련이 있는 데이터의 부분들을 디스플레이하는 것만 필요함을 의미한다. 예를 들어, 모든 XML 데이터가 문서에 디스플레이될 수 있을 것이지만, 다른 한 애플리케이션만이 데이터 내의 하나의 XML 노드의 값들에 관심이 있을 수 있으며, 그러므로 컨텍스트를 보장하기 위해 XML 구조의 나머지 부분에 '관심을 가짐'이 없이 데이터의 그 부분을 디스플레이하는 것만 필요하다.

[0060] 사용자는 동일한 XML 정보를 디스플레이하는 어떤 애플리케이션에 있는 데이터든 편집할 수 있고 데이터의 그 부분을 참조하고 있는 모든 위치들에서 그 데이터를 즉시 업데이트(적용 가능한 비즈니스 논리에 따라)시킨다. 이 능력은 그것이 단일의 XML 문서 내의 다양한 편집 요구 조건들의 오버래핑 특성을 반영하는 편집 환경들의 생성을 허용하므로 각각의 XML 변경을 위한 실시간 메시지를 수신하기 위해 도움이 된다.

[0061] 애플리케이션들은 오류 정보를 공유할 수도 있을 것이다. 콘텐츠 오류들의 사용자 정의된 집합이 각각의 데이터 저장소에 저장될 수 있을 것이다. 예를 들어, 비지니스 논리는 <startDate> 노드가 <endDate> 노드에 선행하는 값을 가져야 함을 지시할 수 있을 것이다. 다수의 데이터 소비자들이 그들의 오류들을 집합적으로 공유하게 하기 위해, XML에 있는 노드들의 목록들 + 오류(오류 텍스트 및 이름으로 구성됨)를 저장하는 각각의 XML 저장소 내에 오류 저장소가 포함된다. XML 변경들과 마찬가지로, 한 클라이언트는 오류를 만들 수 있고, 이어서 그 오류 변경은 각각의 클라이언트에게 방송된다. 그와 같이, 다수의 애플리케이션들이 그 데이터의 모든 표현에 대해 공유될 유효성 검사 논리를 위한 단일의 구현에 의존할 수 있다. 다시 말해서, 동일한 논리가 XML 데이터를 디스플레이하고 있는 각각의 애플리케이션에 복제될 필요가 없다.

[0062] 실행취소 저장소가 글로벌 실행취소 저장소(310)일 수 있거나 실행취소 저장소들이 각각의 XML 저장소와 함께 포함될 수도 있을 것이다. 각각의 데이터 소비자의 변경 요청들은 각각의 변경을 모든 관련 있는 변경들과 조합하는 실행취소 저장소(310) 등과 같은 단일의 실행취소 스택으로 연결되어 각각이 유니트로서 실행취소될 수 있게 할 수 있을 것이다. 이것은 모든 클라이언트들이 모든 문서를 알려진 "양호한" 상태로 유지하면서 최종적인 모든 변경의 '실행취소'를 요청하게 한다.

[0063] 데이터의 동기화는 예정된 그룹의 데이터 소비자들로 제한되지 않는다. 다시 말해서, 새로운 데이터 소비자들은 언제든 어떤 XML 데이터 상에든 알리기 위해 등록할 수 있고 다른 모든 클라이언트들처럼 동일한 데이터를 즉시 편집할 수 있다. 예를 들어, 처음에는 단지 외부의 데이터 소비자 1 및 2만 데이터를 공유하고 있을 수 있다. 나중에, 하나 이상의 데이터 소비자들이 데이터 저장소에 등록하고 데이터를 공유하고 있을 수 있다.

[0064] 한 데이터 소비자는 XML 데이터의 소유자로서 행동하고, XML 데이터의 일관된 형식을 유지하고, 요청 데이터 소비자들에게 사본을 제공하며, 데이터 소비자들로부터 데이터에 대한 변경 요청들을 수신하고, 등록된 데이터 소비자들에게 변경 통지들을 보낼 책임이 있다. 한 실시예에 따르면, 데이터 저장소는 소유자로서 행동하고 각각의 데이터 소비자를 위한 모든 업데이트 및 통지들을 취급한다. XML 데이터 저장소는 워드 프로세싱 애플리케

이션, 스프레드시트 애플리케이션, 슬라이드 프레젠테이션 프로그램, 및 기타의 데이터 소비자 등과 같은 상이한 애플리케이션들에게 이용 가능한 한 세트의 인터페이스를 포함한다. 인터페이스들은, 원하는 XML 데이터 피스들을 얻고, 데이터 소비자가 데이터 저장소에 대해 만들고자 하는 변경들을 데이터 저장소에 알리며, 다른 데이터 소비자들에 의해 그 저장소 아이템에 대해 만들어진 변경들에 관한 XML 데이터 저장소로부터의 통지를 수신하기 위해 등록하도록 안내된다.

[0065] 데이터 저장소가 데이터 소비자에게 변경을 알릴 때마다, 데이터 소비자는 아무것도 하지 않고 변경을 수락하거나, 하나 이상의 부차적 변경을 요청하고 변경을 거절할 수 있다. 부차적 변경은 일반적으로 데이터 저장소에 대해 만들어지는 다른 변경들에 응답하여 변경들을 시작하는 논리의 추가를 포함한다. 예를 들어, 지분 조사서를 이용하는 데이터 소비자는 데이터 저장소 내의 <stockSymbol/> 노드가 변경될 때 통지를 수신하고 변경에 응답하여 웹 서비스에게 데이터를 제출하며 데이터 저장소 내에 있는 <stockData/> 서브트리를 업데이트하고자 할 수 있을 것이다.

[0066] 부차적 변경들은 데이터 저장소에 의해 상이하게 취급되는 실행취소(undo)/취소(cancel)의 목적상 오리지널 변경과 함께 번들로 된다. 그들은 부차적 변경들이 XML 데이터 저장소에 아직 커밋되지 않은 변경에 응답하여 요청되므로 상이하게 취급된다.

[0067] 데이터 소비자(314 및/또는 318)는 XML 데이터 저장소(302)에 대해 변경을 요청하고, 그 변경은 변경이 무효하다거나(예를 들어, 제대로 형성되지 못한 XML), 데이터 소비자 내의 어떤 논리에 의해 변경이 거절되었다는 등을 포함하는 상이한 이유로 인해 거절될 수 있다.

[0068] 어떤 데이터 소비자들은 그들 자신의 버전의 데이터를 XML 데이터 저장소(302)로부터 따로 유지된 저장소(320)에 유지할 수 있을 것이다. 이 XML 데이터의 다수의 사본들을 유지하는 것은 사본들이 동기를 벗어날 수 있음(예를 들어, 속성 패널의 'Title'이 문서 속에 디스플레이된 'Title'과 일치하지 않음)을 포함하는 문제들을 일으킬 수 있을 것이다. 이 문제를 해소하기 위해, 세션 동안에 각각의 XML 데이터 피스의 단일의 "원본(master copy)"이 유지된다. 이 원본은 세션 동안에 다수의 데이터 소비자들에 의해 이용된다. 세션이 종료될 때, 데이터의 다른 사본들은 XML 데이터 저장소의 현재의 상태를 반영하도록 업데이트될 수 있을 것이다. 한 실시예에 따르면, 데이터 저장소(302)는 상이한 데이터 저장소로부터의 동일한 아이템들을 통합하고 나중에 각각의 사본을 다시 저장하도록 구성된다. 보편적인 데이터 아이템을 위한 요청이 수신될 때, 데이터 저장소(302)는 그 두개의 데이터 저장소 아이템들을 단일의 부모 노드에 넣고, 각각의 데이터 아이템과 관련이 있는 스키마들을 가져오는 통합된 XSD를 만들며, 이 저장소 아이템을 위한 인터페이스를 데이터 소비자에게 내준다.

[0069] 데이터 저장소(302)는 과도한 재귀를 탐지하고, 탐지했을 때는 데이터 저장소가 정해진 변경에 응답하여 부차 효과의 루프를 탐지하면 자동적 실패를 유발하도록 구성된다. 한 실시예에 따르면, 깊이(depth)별로 16 레벨 또는 총 1000 부차 효과들을 초과하는 루프가 과도한 것으로 간주된다. 데이터 저장소는 XML 데이터 저장소에 의해 변경이 구조적으로 무효한 것으로 밝혀질 때 어떤 변경 및 부차 효과든 자동으로 거절하도록 구성될 수도 있을 것이다. 이것은 클라이언트에 의해 구조적 변경이 요청되고 구조적으로 무효한 것으로 밝혀지면 데이터 저장소가 자신을 최종적으로 알려진 양호한 상태로 다시 저장하고 다른 데이터 소비자들에게 전달될 수 있을 오류를 생성함을 의미한다.

[0070] 각각의 데이터 소비자(내부 314/외부 318)는 무효한 변경들을 거절할 수도 있을 것이다. 예를 들어, 데이터 소비자는 그 자신의 유효성 검사 계층을 포함할 수 있을 것이다. 무효한 변경이 데이터 소비자 내에 요청되면, 그 변경은 그것의 기존의 유효성 검사 계층에 의해 거절되고, XML 데이터 저장소에 알림이 없이 그 자신의 데이터 저장소의 밖으로 롤백될 수 있을 것이다. 그 변경이 XML 데이터 저장소로부터 생긴 것이면, 데이터 소비자는 데이터 저장소에 의해 건네진 이벤트에 대한 거절을 반환하고, 데이터 저장소는 그것의 '최종적으로 알려진 양호한' 상태로의 취소를 시작한다.

[0071] 데이터 소비자들에 의해 데이터 저장소에 요청된 변경들인 경우에, 데이터 저장소는 그것이 현재의 데이터와 관련이 있는 XML 스키마 집합(305)을 가졌으면 그 변경들의 유효성을 검사하고자 시도한다. 스키마들이 존재하면, 데이터 저장소는 어떤 구조적으로 무효한 것들이든 거절한다.

[0072] 데이터 바인딩을 지원하기 위해, 내부의 데이터 소비자 1(314)과 같은 내부의 애플리케이션 데이터 소비자가 XML 데이터 저장소 및 문서 표면(315) 상에 있는 동작들 사이의 상호작용을 취급한다. 사용자가 데이터 바인딩된 필드를 편집할 때, 그 변경은 문서의 콘텐츠에 영향을 주지만(그래서 애플리케이션의 실행취소 스택에 대해 레코드(record)를 추가함), 데이터 저장소의 XML 콘텐츠에도 영향을 준다(그래서, 데이터 저장소의 실행취소 스

택에 레코드를 추가함). 표면 및 데이터가 항상 동기 상태로 유지되는 것을 보장하는 것을 돕기 위해, 애플리케이션의 실행취소 스택(사용자가 상호작용하는 것)은, 각각의 스택의 상단 아이템을 실행취소하는 것이 애플리케이션 및 데이터 저장소를 동일한 상태로 유지하는 것을 보장하는, 대응하는 XML 데이터 저장소 실행취소 참조(undo reference)에 따라, 표면 변경들을 하나의 실행취소 레코드로 '번들화' 할 수 있다.

[0073] 사용자 시작 실행취소를 취급하기 위해 호스트 애플리케이션을 포함하는 각각의 데이터 소비자를 위해 별도의 실행취소 스택들을 유지하고, 글로벌 실행취소 스택을 공유하며, 현재의 포커스에 기반하여 데이터 소비자를 위한 실행취소를 제한하는 것을 포함하는 다양한 대안들이 이용 가능하다.

[0074] 글로벌 실행취소 스택이 이용될 때, 데이터 소비자는 실행취소 요청을 통해 직접 호스트 애플리케이션으로 가며, 그것은 그것의 실행취소 스택을 떠나 최종적인 아이템을 취한다(실행취소는 포커스가 애프 프레임(app frame)의 어디에 있었는지에 무관하게 동일할 것임). 이것은 XML 데이터 저장소에 대한 모든 변경들이 어떤 일반적인 레코드(예를 들어 "Undo Property Edit")와 함께 호스트의 실행취소 스택에 집중됨을 암시한다. 예를 들어, 사용자가 워드 프로세싱 애플리케이션의 <company/> 필드에 "MICROSOFT Corp."이라고 타이프하면, 그 조작은 데이터 저장소의 실행취소 스택이 그 동작을 포함하게 한다. 다음에, 사용자가 그 텍스트를 제거하기 위해 워드 프로세싱 애플리케이션에서 실행취소를 누르고자 했다면, 워드 프로세싱 애플리케이션은 그 실행취소 스택 상의 최종적인 조작을 실행취소하고(그리고 그 스택 상의 최종적인 연산을 실행취소하도록 데이터 저장소에 알림), 그것은 다른 클라이언트가 그 동작을 하는 것으로 귀결될 것이다. 반대로, 사용자가 워드 프로세싱 애플리케이션에 바인딩되지 않는 어떤 텍스트를 타이프하고 패널에서 실행취소를 눌렀으면, 다른 클라이언트는 호스트를 통해 그 요청을 버릴 것이고, 그것은 그것의 실행취소 스택(이 경우에, 문서 표면에 대한 편집)으로부터 최종적인 동작을 제거할 것이다.

[0075] 데이터 소비자가 XML 데이터 저장소에 의해 보내진 변경을 거절할 때, XML 데이터는 '불량한' 비지니스 논리 상태로 종료될 수 있다. 예를 들어, 비용 보고서에 대한 검사를 수행하는 비지니스 논리가 있다고 가정하자. 논리는, 한 라인 아이템이 $100을 초과하는지를 검사하고, 만일 그렇다면, 데이터 소비자는 <lineItemAmount/>의 업데이트를 거절하는 것을 포함한다. 그렇지 않다면, 데이터 소비자는 새로운 라인 아이템 값으로 총액을 업데이트한다. 총액이 $500을 초과하면, 데이터 소비자는 <reportTotal/>의 업데이트를 거절한다. 위 논리를 이용하여, 사용자가 송장 라인(invoice line)에 총액이 $500을 초과하게 하는 $50을 입력한다고 가정하면, 제1 논리 검사는 성공하지만, 제2 논리 검사는 총액의 업데이트를 거절한다. 이것은 방금 최종적인 변경이 실행취소되었다면, 라인 아이템들의 합계가 총액과 일치하지 않는 송장이 있을 것임을 의미한다. 결과적으로, 한 실시예에 따르면 오리지널 변경의 모든 부차 효과들이 실행취소된다.

[0076] 데이터 저장소(302)는 실행취소 목적상 이러한 트랜잭션들을 서로 번들링하는 것을 허용하는 트랜잭션 메커니즘으로서 행동한다. 실시예에 따르면, '거절' 을 취급하기 위한 3개의 상이한 대안들이 이용된다. 우선, 데이터 저장소는 유효한 상태로 되돌리기 위해 변경 실행취소를 내보낼 수 있을 것이다('롤백' 이 호출되기도 함). 두번째로, 실행취소(undo) 및 취소(calcel)는 아무런 패리티(parity)도 갖지 않으며, 세번째로, 어떤 클라이언트도 취소(cancel)할 수 없다.

[0077] 제1 대안은 XML 데이터 저장소가 유효한 상태로 되돌아가도록 변경 실행취소를 내보내게 하려는 것이다. 이것은 사실상 모든 조작을 실행취소하고 비지니스 논리 오류를 트리거했던 변경으로 되돌린다. 이 대안에서, XML 데이터 저장소는 TRUE로 설정된 '실행취소' 플래그를 갖는 변경 요청들을 내보낼 수 있고, 데이터 소비자가 이러한 변경을 그 자신에게 수행하게 하며, XML 데이터 저장소는 그 '실행취소' 플래그가 설정된 변경 요청들을 내보낼 수 있다. 다음은 한 예이다.

```
User edits node A in a Client 1{
    (Client 1 does internal logic)
    Tell XML Data Store{
        Store Updates
        Tell Client 2 about A{
            Change B{
                Queue it up, return OK
            }
        }
        Tell Client 1 about B{
            Internal Logic
                Change C{
                    Internal Logic
                    Tell XML Data Store{
                        Queue it up, return OK
                    }
                    OAC
                }
        }
        Tell Client 2 about C{
            Change D{
                Queue it up, return OK
                }
            }
        Tell Client 1 about D{
            Internal Logic
                **REJECTION**
            Rollback DOM change to D
            Return FAIL
        }
        Tell Client 2 to undo D
        Tell Cleint 2 to undo C
        Tell Client 1 to undo C – *undo while exiting after a rejection*
        Tell Client 1 to undo B – *undo while exiting after a rejection*
        Tell Client 2 to undo B
        Tell Client 2 to undo A
        Return FAIL to initial change request
    }
}
```

[0078]

[0079] 사실상, 데이터 소비자는 두개의 XML 데이터 저장소 호출(부모 변경 A가 존재하는 동안 변경 B 및 C를 실행취소하는 요청)들을 수락하거나, 또는 두개의 XML 문서 개체 관리자들이 동기를 벗어난다.

[0080] 다른 한 대안은, 데이터 소비자가 현재의 변경을 '취소(cancel)'시키는 것이 솔루션을 무효한 비지니스 논리 상태로 할 수 있는 한, 취소(cancel)와 실행취소(undo) 사이의 현재의 디스패리티(disparity)를 유지하려는 것이다. 이 경우에, XML 데이터 저장소는 그 변경을 실행취소하여 거절(reject)/취소(cancel)되었던 변경*만*을 롤백하지만, 저장소의 실행취소 스택에 있는 다른 어떤 변경들은 실행취소하지 않고 중지할 수 있을 것이다.

[0081] 데이터 소비자들은 그들 자신의 고유의 실행취소 스택들을 유지할 수도 있을 것이다. 그러나, 애플리케이션들은 그들의 스택의 최상위 동작이 라이브 동기화 경계의 다른 쪽에서 요청된 조작과 일치하지 않을 때 그들의 실행취소 스택에 추가적 동작을 추가하지 않는 것에 관해 '스마트' 해야 한다. 예를 들어, 사용자가 워드 프로세싱 애플리케이션의 <company/> 필드에 "MICROSOFT Corp."라고 타이프하면, 그 조작은 '라이브 동기화'되고, 각각의 등록된 데이터 소비자의 실행취소 스택 상에 보이는 실행취소 동작으로 귀결된다. 다음에, 사용자가 방금 입력된 텍스트를 제거하기 위해 워드 프로세싱 애플리케이션에서 실행취소를 누르고자 했으면, 워드 프로세

싱 애플리케이션은 그 실행취소 스택 상의 최종 조작을 실행취소할 것이지만(그리고, 데이터 저장소에게 그 스택 상의 최종 조작을 실행취소하라고 말함), 저장소의 실행취소 스택은 데이터 저장소와 반드시 일치해야 하기 때문에, 이 때 데이터 소비자는 이 요청을 알고 그 실행취소 스택을 떠나 최종 동작을 취한다(어떤 부차 효과든 포함함). 그 후, 사용자가 데이터 소비자에서 실행취소를 눌렀다면, 데이터 소비자는 데이터 저장소로부터 실행취소 동작을 얻을 때 그 실행취소 스택 및 워드 프로세싱 애플리케이션으로부터 최종 동작을 제거하고, 그것이 그 실행취소 스택 상의 최종 동작과 일치하는지를 주목하며, 또한 그것을 제거할 것이다. 반대로, 사용자가 워드 프로세싱 애플리케이션에 바인딩되지 않는 텍스트를 타이프했다면(워드 프로세싱 애플리케이션에 대해 실행취소 레코드들을 추가함), 그들이 데이터 소비자에서 실행취소를 누르기 전에, 클라이언트는 그 실행취소 스택으로부터 최종 동작을 제거하고, 저장소는 동일하게 하지만, 워드 프로세싱 애플리케이션은 다른 한 동작을 추가한다(실행취소 스택 상의 최종의 것은 최종 XML 변경이 아니기 때문). 그러나, 워드 프로세싱 애플리케이션은 그 오리지널 상태로 되돌아가기 위해 재실행을 수행해야 한다는 사실에 따라 그 변경을 저장할 것이다.

[0082] 다른 한 대안은 데이터 소비자들이 서로를 인식하게 하는 것을 포함한다. 실행취소 컨트롤은 데이터 소비자들 사이에 전달될 수 있다. 예를 들어, 사용자가 나중에 다른 한 데이터 소비자에게 방송될 한 애플리케이션의 <company/> 필드를 편집하면, 실행취소 스택은 아래와 같이 보일 것이다.

[0083]

Store	Application	Data consumer 1
Marker (pass to Data Consumer 1)	Company → "Microsoft Corp." + store undo	Company → "Microsoft Corp"

[0084] 이 경우에 XML 데이터 저장소 상의 실행취소 레코드는 트랜잭션을 호스트하지 않을 것이다. 오히려, 컨트롤을 클라이언트에게 전달하여 실행취소 동작을 완료하게 한 마커(marker)를 호스트할 것이다. 사용자가 워드 프로세싱 애플리케이션으로 가서 실행취소를 수행했다고 가정하면, 애플리케이션의 캔버스는 뒤로 업데이트하고, 컨트롤을 데이터 저장소로 전달할 것이다. XML 데이터 저장소는 최종 트랜잭션을 실행취소하고자 시도할 것이지만, 마커를 보고 그 실행취소를 위해 컨트롤을 데이터 소비자에게 전달한다. 데이터 소비자는 저장소에 대한 실행취소 요청을 수행할 것이고, 나중에 그것을 다른 모든 클라이언트들에게 실행취소로서 방송할 것이다. 이것은 사용자가 클라이언트에서 동작을 수행했으면, 워드 프로세싱 애플리케이션의 상이한 필드에서의 동작에 따라, 실행취소 스택들이 아래와 같이 보일 것임을 의미한다.

[0085]

Store	Application	Data consumer 1
Marker (pass to Process 1)	Company → "Microsoft Corp." + store undo	Company → "Microsoft Corp"
Date → "January 20, 2005"	Date → "January 29, 2005" + store undo	Marker (pass to host)

[0086] 이 경우에, 사용자의 다음 동작이 호스트 애플리케이션에서의 실행취소이었으면, 호스트(그리고 데이터 저장소)는 데이터 소비자가 그 자신의 DOM 상에서 수행하고 실행취소 마커를 버릴 것인 실행취소 동작을 할 것이다. 사용자의 다음 동작이 데이터 소비자에서의 실행취소이었으면, 데이터 소비자는 호스트에게 그 동작을 위한 컨트롤을 주고(상단 동작이 단지 '마커'이기 때문에) 동일한 호스트 실행취소가 수행될 것이다.

[0087] 도 4는 라이브 동기화의 예를 예시한다. 가장 단순한 한 경우에, '라이브 동기화'는 다른 한 위치/애플리케이션의 UI에 실시간으로 반영될 한 위치(예를 들어, 워드 프로세서 애플리케이션)에 존재하는 XML 데이터에 대한 사용자 편집을 위한 능력을 말한다. 문서가 편집되는 동안, XML 데이터에 영향을 주는 변경들은 워드 프로세싱 애플리케이션 및 속성 패널 등과 같은 데이터에 관심이 있는 다른 등록된 데이터 소비자들에게 전달된다. 이것은 각각의 애플리케이션의 XML 데이터의 콘텐츠가 동일하게 유지되는 것을 보장하는 것을 돕는다.

[0088] 창(400)을 보면, 문서(415)가 편집을 위해 열려 있고 속성 패널(420)이 보여지고 있다. 속성 패널(405) 및 문서(410) 모두에 타이틀이 보여진다. 창(400)의 타이틀(405)이 창(425) 내의 타이틀(435) 및 타이틀(440)에 예시된 바와 같이 Data Binding - Live Sync Integration로부터 Foo Bar Biz로 변경되는 것을 가정하자. 속성 패널 타이틀(435) 내에서 타이틀이 업데이트되자마자, 변경이 변경을 수락 또는 거절할 수 있는 워드 프로세싱 애플리케이션으로 보내진다. 이 예에서, 문서 내의 속성 패널 애플리케이션 및 타이틀(440)을 이용하여 이루어진 타이틀에 대한 변경을 수락한 애플리케이션이 업데이트된다.

[0089] 도 5 내지 8은 데이터 소비자들 사이의 XML 데이터의 실시간 동기화를 위한 프로세스들을 예시한다.

[0090] 여기에 있는 루틴들의 설명을 읽을 때, 다양한 실시예들의 논리적 조작들이 (1) 컴퓨팅 시스템 상에서 실행되는 컴퓨터 구현 동작 또는 프로그램 모듈들의 시퀀스로서 및/또는 (2) 컴퓨팅 시스템 내의 상호접속된 머신 논리 회로 또는 회로 모듈들로서 구현됨을 알수 있을 것이다. 구현은 본 발명을 구현하는 컴퓨팅 시스템의 성능 요구조건들에 따른 선택의 문제이다. 따라서, 여기에 기술된 실시예들을 이루고 예시된 논리적 연산들은 연산, 구조적 장치, 동작 또는 모듈들이라고 다양하게 지칭된다. 이러한 연산, 구조적 장치, 동작 및 모듈들은 소프트웨어, 펌웨어, 전용 디지털 논리 및 그들의 어떤 조합으로든 구현될 수 있을 것이다.

[0091] 도 5는 두개의 클라이언트들과 XML 데이터 저장소 사이의 상호작용을 예시한다.

[0092] 데이터 저장소(520)는 애플리케이션(510)을 이용하여 만들어지는 노드 A에 대한 사용자 편집을 수신한다. 데이터 소비자 1(530)은 데이터 저장소(520)로부터 노드 A에 대한 변경에 관한 통지를 수신한다. 노드 A에 대한 변경의 결과로서, 데이터 소비자 1은 노드 B에 대한 부차적 변경을 요청한다. 데이터 저장소(520)는 나중의 실행을 위해 부차적 변경 B를 대기시킨다. 모든 부차적 변경들이 데이터 소비자 1로부터 대기되면, 데이터 저장소(520)는 데이터 소비자 2(540)에게 노드 A에 대한 변경을 알린다. 데이터 소비자 2는 노드 C에 대한 부차적 변경을 요청한다. 응답으로, 데이터 저장소는 나중의 실행을 위해 노드 C에 대한 변경을 대기시킨다. 이 시점에서, 노드 A에 관한 처리가 완료되지만, 노드 B 및 노드 C에 대한 부차적 변경들은 여전히 미정으로 남아 있다. 데이터 소비자 1이 노드 B에 대한 변경을 요청했으므로, 데이터 저장소는 데이터 소비자 2에게 제안된 B의 변경에 관한 통지를 보낸다. 데이터 소비자 2는 응답으로 어떤 변경도 만들지 않으며 변경을 수락한다. 마찬가지로, 데이터 소비자 2가 노드 C에 대한 변경을 요청했으므로, 데이터 저장소는 데이터 소비자 1에게 노드 C의 변경에 관한 통지를 보낸다. 데이터 소비자 1은 변경을 수락한다. 이 예에서, 관심 있는 데이터 소비자들 모두에 의해 모든 변경들이 수락되었다. 그러므로, 데이터 저장소는 데이터 저장소에게 변경을 커밋한다. 간단히 말해서, 데이터 저장소는 각각의 데이터 소비자가 변경에 응답하여 어떤 변경이든 수행하게 하고, 데이터 저장소는 이러한 변경들을 수신된 순서대로 실행하고 메시지를 보냄으로써 그들에게 연속으로 메시지가 보내지게 한다.

[0093] 도 6은 두개의 외부의 데이터 소비자들 사이의 상호작용 및 XML 데이터 저장소에 대한 변경을 도시한다.

[0094] 도 5에 도시된 문서를 만든 애플리케이션 등과 같은 내부의 클라이언트에 의해 오리지널 변경이 발생된 경우에, 데이터 소비자는 거의 변경되지 않는다. 이 예는 다수의 클라이언트들로부터의 두가지 기본적인 변경 규칙들을 예시한다. 제1 규칙은 상단 레벨 변경들이 우선 깊이(depth)에서 일어나는 것이다. 제2 규칙은 부차적 변경들을 대기시키는 것을 포함한다.

[0095] 제1 규칙은 변경의 부차 효과들이 어떤 새로운 변경이든 일어날 수 있기 전에 일어난다는 사실에 관한 것이다. 데이터 소비자 1이 두개의 변경들을 만들기 위한 기능을 실행하는 다음의 예를 고려하자. 제1 변경은 노드 A에 대해 만들어지고 제2 변경은 노드 B에 대해 만들어진다.

[0096] 데이터 소비자 1(630)은 노드 A에 대한 변경을 요청하고, XML 데이터 저장소(620)가 노드 A의 변경에 대한 요청을 수신한 후 통지가 데이터 소비자 2(640)에게 보내진다. 응답으로, 데이터 소비자 2는, 나중의 실행을 위해 데이터 저장소에 의해 대기되는, 노드 C에 대한 부차적 변경을 요청한다. 그 후, 데이터 소비자 1은, 노드 A에 대한 변경이 아직 완료되지 않았으므로 대기되는, 노드 B에 대한 변경을 요청한다. 데이터 소비자 2는 노드 A에 대한 변경을 수락하고, 응답으로, 데이터 저장소는 대기된 C에 대한 부차적 변경을 실행한다. 데이터 소비자 1은 노드 C에 대한 변경의 통지를 수신하고 변경에 응답할 수 있을 것이다. 이 예에서는, 데이터 소비자 1이 변경을 수락한다. 그 후, 데이터 저장소는, 데이터 소비자 1에 의해 요청되었던, 대기된 B에 대한 부차적 변경을 실행했다. 데이터 소비자 2는 데이터 저장소로부터, 그것이 응답할 수 있을 것인, B의 변경에 관한 통지를 수신한다. 데이터 소비자 2는 변경을 수락하고 변경들이 데이터 저장소에 커밋된다.

[0097] 이 경우에, 데이터 소비자 2는 다음의 두개의 코드 라인들이 두개의 고유의 변경들을 트리거하도록 우선 A에 대한 변경에 응답한다.

```
…
doc.CustomXMLParts(1).SelectSingleNode().AddNode(“foo”,”bar”)
doc. CustomXMLParts (1).SelectSingleNode().AddNode(“foo2”,”bar”)
```

[0098] …

[0099] 제2 규칙은 변경의 부차 효과들이 그들이 요청됨에 따라 대기된다는 사실에 관한 것이고, 그러므로 다수의 변경들이 제1 변경의 부차 효과들이 일어나기 전에 대기될 수 있을 것이다.

[0100] 도 7은 다수의 부차적 변경들을 포함하는 프로세스를 도시한다. 다음의 예는 다음의 변경들을 예시한다. 데이터 소비자 1(730)이 노드 A가 변경되었다는 통지를 수신할 때, 데이터 소비자 1(730)은 노드 B 및 C에 대한 부차적 변경들을 수행하고자 한다고 가정하자. 데이터 소비자 2(740)는 노드 B에 대한 변경에 관한 통지에 응답하여 노드 D, E, 및 F에 대한 부차적 변경들을 수행하고자 한다.

[0101] 도 7을 보면, 노드 B의 변경에 관한 부차 효과들이 실행되기 전에 데이터 소비자 1이 노드 A에 대한 변경에 관한 통지의 결과로서의 그 부차적 변경들의 전부를 유발하는 것을 알 수 있다. 그러므로, 노드 C에 대한 변경들은 노드 D, E, 및 F에 대한 변경에 앞서 일어난다.

[0102] 도 8은 콜러의 부차 효과들이 최후에 실행되는 것을 도시하는 프로세스를 예시한다. 이 경우에, 데이터 소비자 1 자신에 의해 발생된 부차 효과들은 다른 모든 클라이언트들이 변경을 알고 그들 자신의 부차 효과들을 발생시킬 기회를 가진 후에 일어날 것임을 알 수 있다. 이것은 XML 데이터 저장소가 그것이 성공 조건을 콜러에게 반환하고, 콜러가 부차 효과들을 일으킬 이벤트를 제공하게 할 수 있기 전에 각각의 클라이언트에게 변경을 알린다는 사실(각각의 클라이언트가 그 변경을 수락/거절하기 위한 기회를 갖는 것을 보장하기 위해)의 필수적인 부차 효과이다. XML 데이터 저장소의 모든 클라이언트들에 의해 변경이 수락 또는 거절되는지를 듣기 위해 변경을 요청한 데이터 소비자가 최종이어야 함을 예견할 수 있다. 또한, 이것은 두개의 변경들이 저촉하고 있고 구조적이지 못할지라도, 일반적으로 원하는 결과인 콜러의 변경이 얻어지는 것을 보장하는 것에 도움이 된다.

[0103] 위 명세서, 예 및 데이터는 본 발명의 구성의 제조 및 이용에 관한 완전한 기술을 제공한다.

도면의 간단한 설명

[0006] 도 1은 컴퓨터를 위한 예시적 컴퓨팅 아키텍처를 예시한다.

[0007] 도 2는 하나 이상의 클라이언트 애플리케이션과 하나 이상의 데이터 저장소 및 데이터 저장소의 콘텐츠 사이의 관계를 예시하는 블록 다이어그램이다.

[0008] 도 3은 XML 데이터 저장소들에 의한 내부 및 외부의 데이터 소비자들 사이의 상호작용을 도시하는 시스템 다이어그램을 예시한다.

[0009] 도 4는 라이브 동기화의 예를 예시한다.

[0010] 도 5는 두개의 클라이언트와 XML 데이터 저장소 사이의 상호작용을 예시한다.

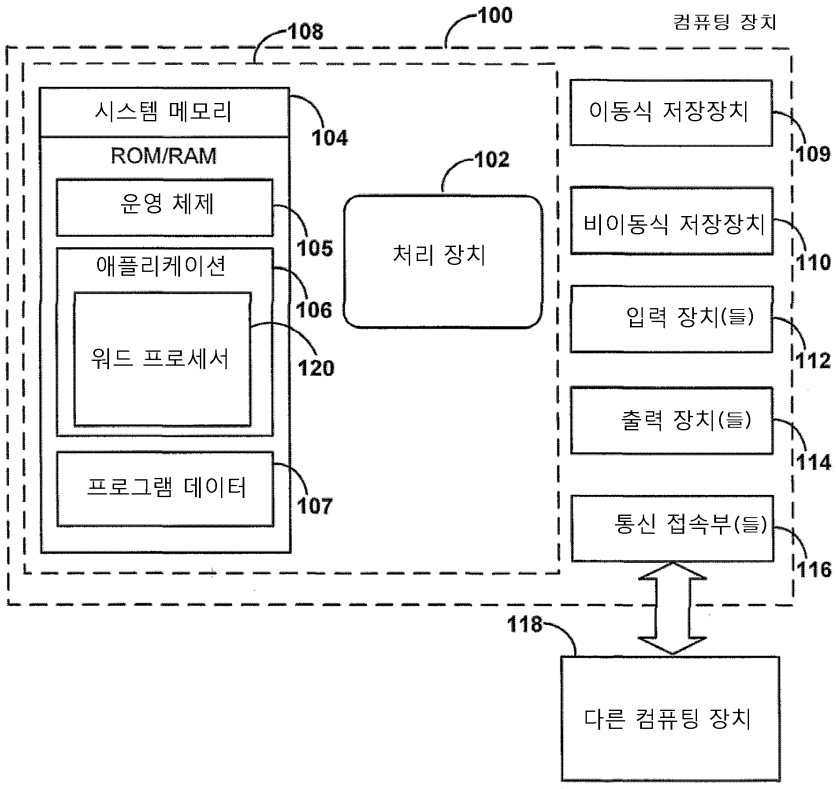
[0011] 도 6은 두개의 외부의 데이터 소비자들과 XML 데이터 저장소에 대한 변경 사이의 상호작용을 도시한다.

[0012] 도 7은 다수의 부차적 변경들을 포함하는 프로세스를 도시한다.

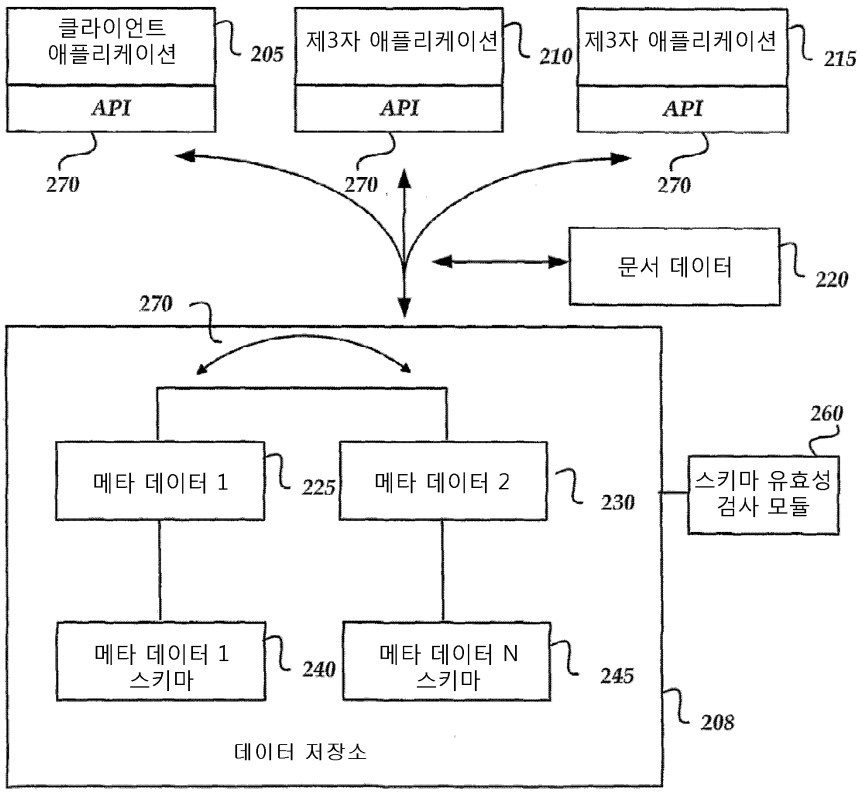
[0013] 도 8은 본 발명의 양태들에 따라 콜러의 부차 효과들이 최후에 실행되는 것을 도시하는 프로세스를 예시한다.

도면

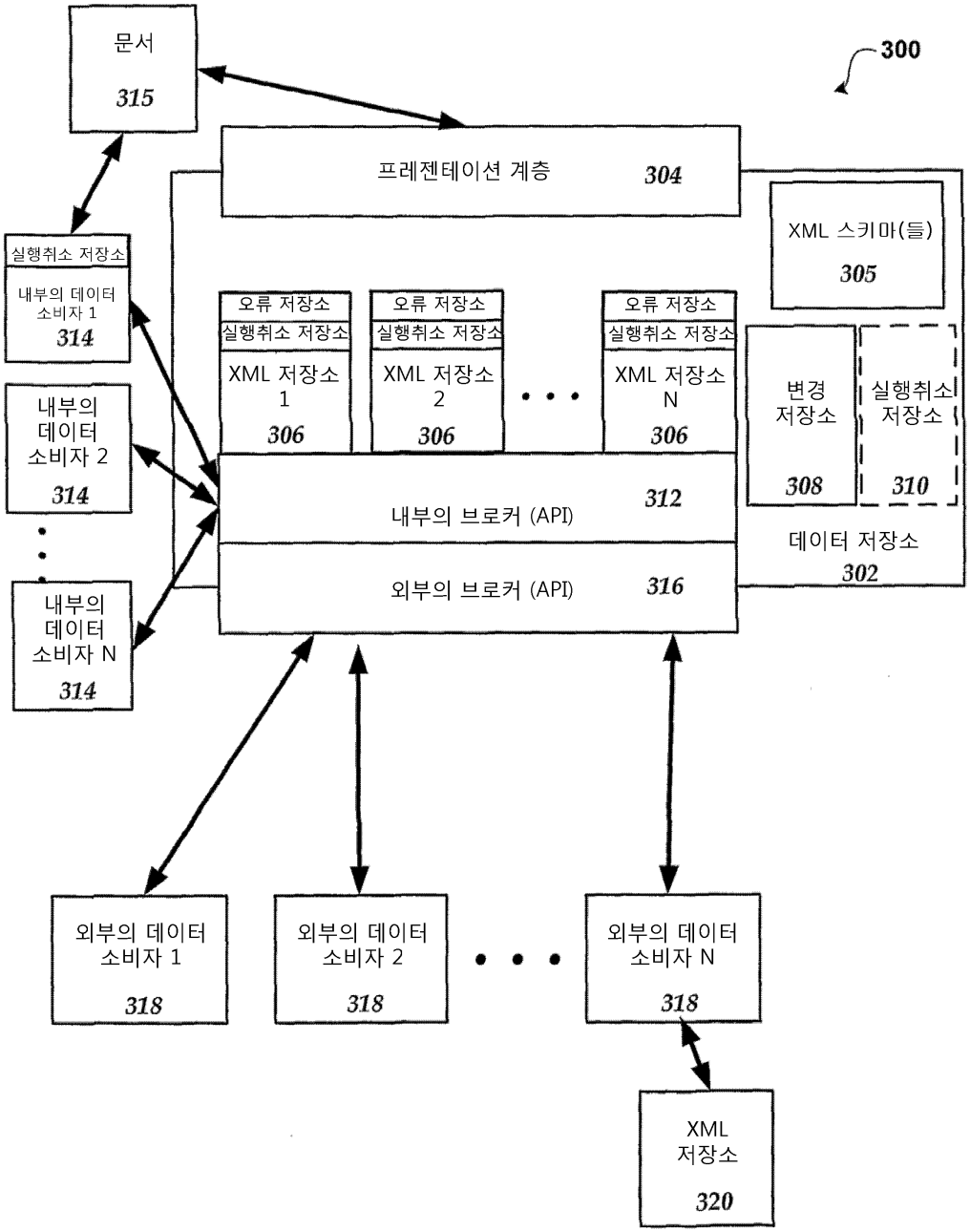
도면1


100
컴퓨팅 장치
108
시스템 메모리
104
ROM/RAM
운영 체제
105
애플리케이션
106
워드 프로세서
120
프로그램 데이터
107
102
처리 장치
이동식 저장장치
109
비이동식 저장장치
110
입력 장치(들)
112
출력 장치(들)
114
통신 접속부(들)
116
118
다른 컴퓨팅 장치

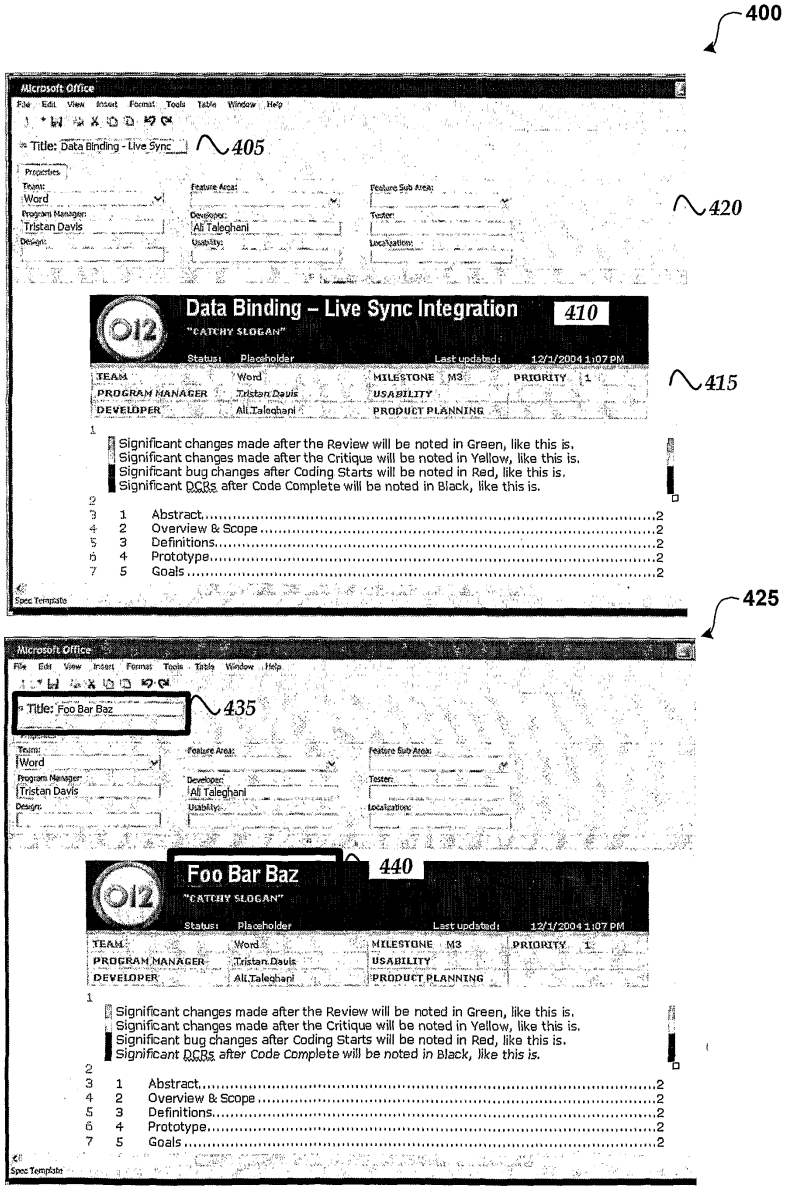
도면2


클라이언트 애플리케이션
205
제3자 애플리케이션
210
제3자 애플리케이션
215
API
API
API
270
270
270
문서 데이터
220
270
260
메타 데이터 1
225
메타 데이터 2
230
스키마 유효성 검사 모듈
메타 데이터 1 스키마
240
메타 데이터 N 스키마
245
208
데이터 저장소

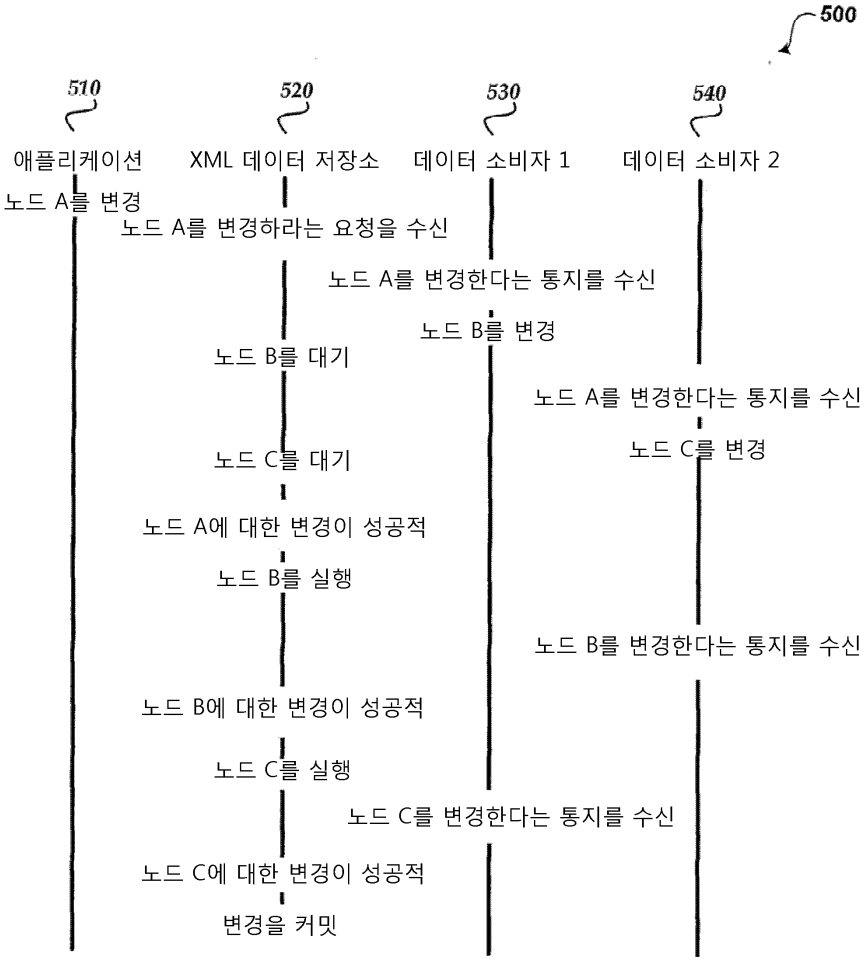
도면3


문서
315
300
프레젠테이션 계층
304
XML 스키마(들)
305
실행취소 저장소
내부의 데이터 소비자 1
314
오류 저장소
실행취소 저장소
XML 저장소 1
306
오류 저장소
실행취소 저장소
XML 저장소 2
306
오류 저장소
실행취소 저장소
XML 저장소 N
306
변경 저장소
308
실행취소 저장소
310
내부의 데이터 소비자 2
314
내부의 브로커 (API)
312
외부의 브로커 (API)
316
데이터 저장소
302
내부의 데이터 소비자 N
314
외부의 데이터 소비자 1
318
외부의 데이터 소비자 2
318
외부의 데이터 소비자 N
318
XML 저장소
320

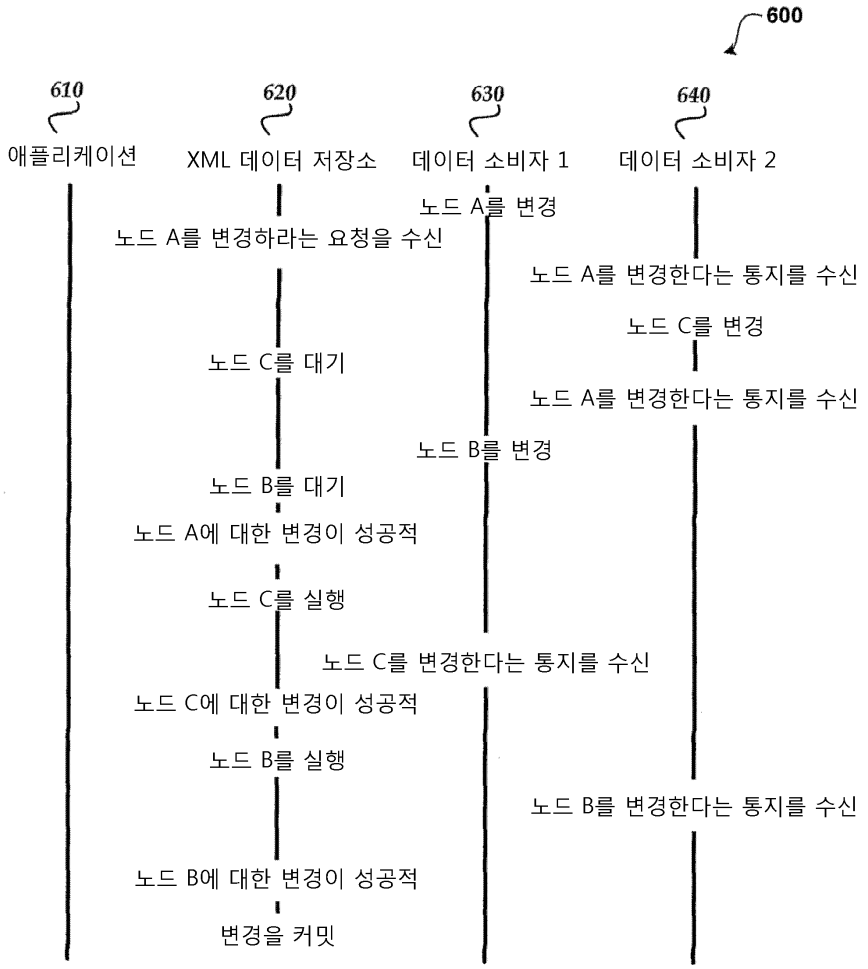
도면4


400
Microsoft Office
Title: Data Binding - Live Sync
405
Word
Tristan Davis
Ali Taleghani
420
Data Binding – Live Sync Integration
410
"CATCHY SLOGAN"
Status: Placeholder
Last updated: 12/1/2004 1:07 PM
TEAM Word
MILESTONE M3
PRIORITY 1
PROGRAM MANAGER Tristan Davis
USABILITY
DEVELOPER Ali Taleghani
PRODUCT PLANNING
415
Significant changes made after the Review will be noted in Green, like this is.
Significant changes made after the Critique will be noted in Yellow, like this is.
Significant bug changes after Coding Starts will be noted in Red, like this is.
Significant DCRs after Code Complete will be noted in Black, like this is.
1 Abstract 2
2 Overview & Scope 2
3 Definitions 2
4 Prototype 2
5 Goals 2
Spec Template
425
Microsoft Office
Title: Foo Bar Baz
435
Word
Tristan Davis
Ali Taleghani
Foo Bar Baz
440
"CATCHY SLOGAN"
Status: Placeholder
Last updated: 12/1/2004 1:07 PM
TEAM Word
MILESTONE M3
PRIORITY 1
PROGRAM MANAGER Tristan Davis
USABILITY
DEVELOPER Ali Taleghani
PRODUCT PLANNING
Significant changes made after the Review will be noted in Green, like this is.
Significant changes made after the Critique will be noted in Yellow, like this is.
Significant bug changes after Coding Starts will be noted in Red, like this is.
Significant DCRs after Code Complete will be noted in Black, like this is.
1 Abstract 2
2 Overview & Scope 2
3 Definitions 2
4 Prototype 2
5 Goals 2
Spec Template

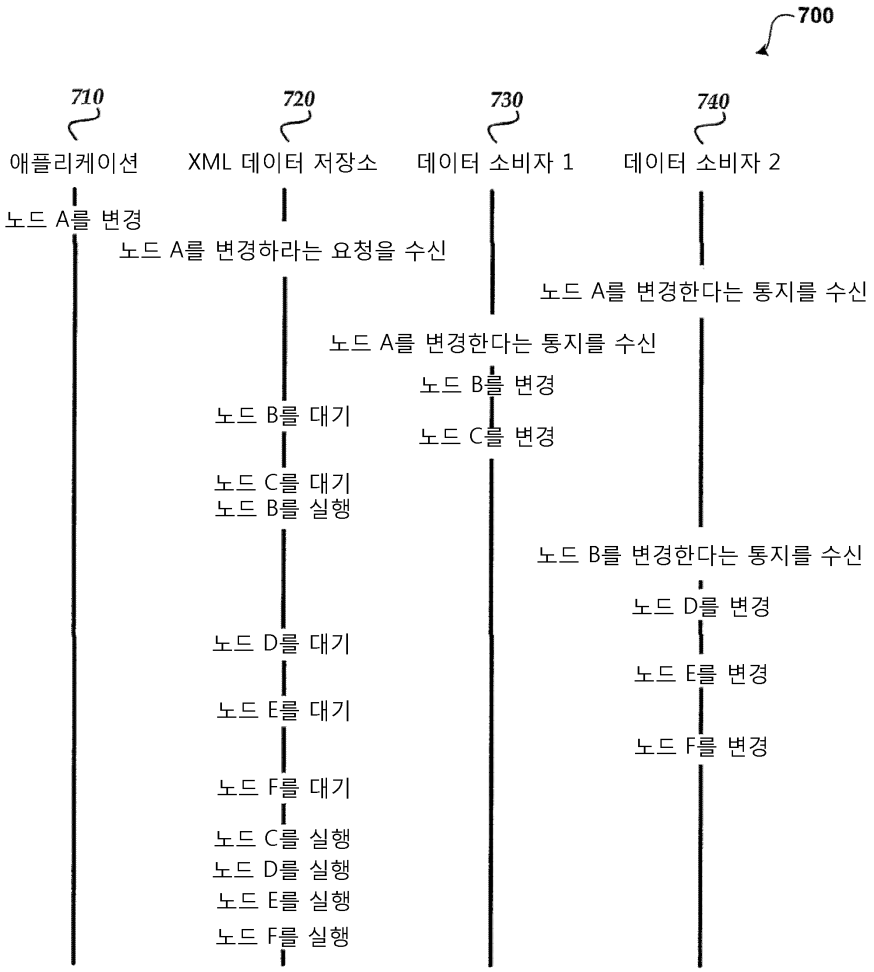
도면5


500
510
520
530
540
애플리케이션
XML 데이터 저장소
데이터 소비자 1
데이터 소비자 2
노드 A를 변경
노드 A를 변경하라는 요청을 수신
노드 A를 변경한다는 통지를 수신
노드 B를 변경
노드 B를 대기
노드 A를 변경한다는 통지를 수신
노드 C를 변경
노드 C를 대기
노드 A에 대한 변경이 성공적
노드 B를 실행
노드 B를 변경한다는 통지를 수신
노드 B에 대한 변경이 성공적
노드 C를 실행
노드 C를 변경한다는 통지를 수신
노드 C에 대한 변경이 성공적
변경을 커밋

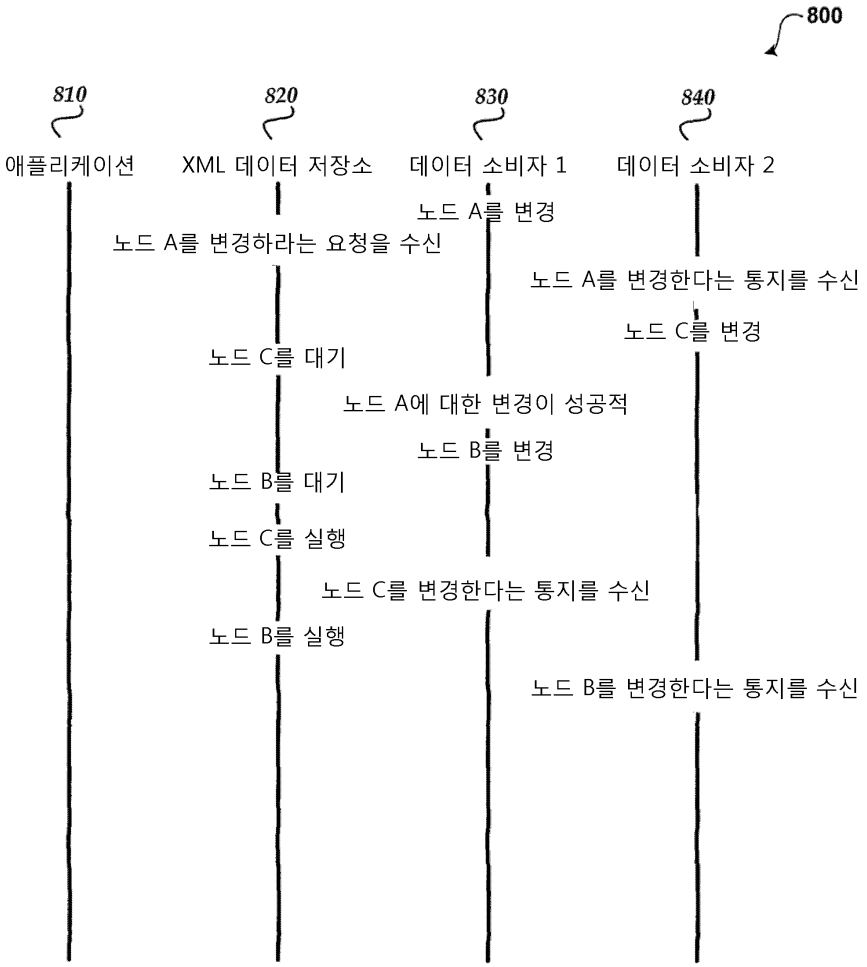
도면6


600
610
620
630
640
애플리케이션
XML 데이터 저장소
데이터 소비자 1
데이터 소비자 2
노드 A를 변경
노드 A를 변경하라는 요청을 수신
노드 A를 변경한다는 통지를 수신
노드 C를 변경
노드 C를 대기
노드 A를 변경한다는 통지를 수신
노드 B를 변경
노드 B를 대기
노드 A에 대한 변경이 성공적
노드 C를 실행
노드 C를 변경한다는 통지를 수신
노드 C에 대한 변경이 성공적
노드 B를 실행
노드 B를 변경한다는 통지를 수신
노드 B에 대한 변경이 성공적
변경을 커밋

도면7


700
710
720
730
740
애플리케이션
XML 데이터 저장소
데이터 소비자 1
데이터 소비자 2
노드 A를 변경
노드 A를 변경하라는 요청을 수신
노드 A를 변경한다는 통지를 수신
노드 A를 변경한다는 통지를 수신
노드 B를 변경
노드 B를 대기
노드 C를 변경
노드 C를 대기
노드 B를 실행
노드 B를 변경한다는 통지를 수신
노드 D를 변경
노드 D를 대기
노드 E를 변경
노드 E를 대기
노드 F를 변경
노드 F를 대기
노드 C를 실행
노드 D를 실행
노드 E를 실행
노드 F를 실행

도면8


800
810
820
830
840
애플리케이션
XML 데이터 저장소
데이터 소비자 1
데이터 소비자 2
노드 A를 변경
노드 A를 변경하라는 요청을 수신
노드 A를 변경한다는 통지를 수신
노드 C를 변경
노드 C를 대기
노드 A에 대한 변경이 성공적
노드 B를 변경
노드 B를 대기
노드 C를 실행
노드 C를 변경한다는 통지를 수신
노드 B를 실행
노드 B를 변경한다는 통지를 수신