(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2017-0052668
(43) 공개일자 2017년05월12일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2006.01)
(52) CPC특허분류
G06F 11/3684 (2013.01)
G06F 11/3692 (2013.01)
(21) 출원번호 10-2017-7009686
(22) 출원일자(국제) 2015년09월04일
심사청구일자 없음
(85) 번역문제출일자 2017년04월10일
(86) 국제출원번호 PCT/US2015/048528
(87) 국제공개번호 WO 2016/040154
국제공개일자 2016년03월17일
(30) 우선권주장
62/047,256 2014년09월08일 미국(US)

(71) 출원인
아브 이니티오 테크놀로지 엘엘시
미국 02421 매사추세츠주 렉싱턴 스프링 스트리트 201
(72) 발명자
프린츠, 필리프
미국 메사추세츠 01821 빌레리카 콩코드 로드 313
이스만, 마샬 앨런
미국 메사추세츠 02458 뉴튼 밸리스프링 로드 11
(74) 대리인
인비전 특허법인

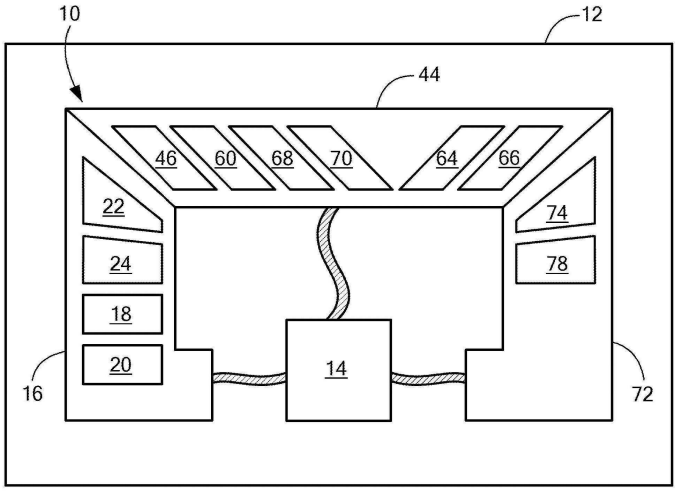
전체 청구항 수 : 총 37 항

(54) 발명의 명칭 **데이터 구동 테스트 프레임워크**

(57) 요 약

어플리케이션을 테스트하기 위한 장치는 메모리 및 상기 메모리에 동작 가능하게 커플링된 프로세서를 포함하는 데이터 처리 머신을 포함한다. 데이터 처리 머신은 데이터 엔지니어링 모듈 (16), 계산 환경 관리자 (44), 그리고 결과 분석 모듈 (72) 을 포함하는 데이터 구동 테스트 프레임워크를 구현하도록 구성된다. 데이터 엔지니어링 모듈은 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터를 생성하도록 구성된다. 계산 환경 관리자는 어플리케이션이 엔지니어링 된 테스트 데이터에 동작할 계산 환경을 제어하도록 구성된다. 결과 분석 모듈은 어플리케이션에 의해 동작한 엔지니어링 된 테스트 데이터를 예상되는 출력과 비교하도록 구성된다.

대 표 도 - 도1



명 세 서

청구범위

청구항 1

어플리케이션들을 테스트하기 위한 장치로서, 상기 장치는 메모리 및 상기 메모리에 동작 가능하게 커플링 된 프로세서를 포함하는 데이터 처리 머신을 포함하고, 상기 데이터 처리 머신은 데이터 엔지니어링 모듈, 계산 환경 관리자, 및 결과 분석 모듈을 포함하는 데이터 구동 테스트 프레임워크를 구현하도록 구성되고,

상기 데이터 엔지니어링 모듈은, 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터를 생성하도록 구성되고,

상기 계산 환경 관리자는, 상기 어플리케이션이 상기 엔지니어링 된 테스트 데이터 상에 동작할 계산 환경을 제어하도록 구성되고,

상기 결과 분석 모듈은, 상기 어플리케이션에 의해 동작되는 엔지니어링 된 테스트 데이터를 예상되는 출력과 비교하도록 구성되는, 어플리케이션을 테스트하기 위한 장치.

청구항 2

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 생산 (production) 데이터의 서브세트를 추출하도록 구성되고,

상기 서브세트는, 명시된 코드 범위 (coverage) 를 달성하도록 선택되고, 그리고

상기 엔지니어링 된 테스트 데이터는, 상기 생산 데이터의 상기 서브세트를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 3

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 생산 데이터로부터 증류된 (distilled) 데이터를 생성하기 위한 데이터 증류기 (still) 를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 4

제 34 항에 있어서,

상기 추가 데이터는 명시된 코드 범위를 달성하도록 선택되는, 어플리케이션을 테스트하기 위한 장치.

청구항 5

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 상기 데이터 증류기로부터 증류된 데이터를 수신하고 상기 증류된 데이터를 강화 (enhance) 하기 위한 데이터 강화기 (enhancer) 를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 6

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 상기 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 데이터를 생성하도록 구성되고, 상기 생성된 데이터는 명시된 코드 범위를 달성하도록 선택되고, 상기 엔지니어링 된 테스트 데이터는 상기 생성된 데이터를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 7

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 포지티브 데이터를 생성하기 위한 포지티브 데이터 제조기를 더 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 8

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 상기 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 데이터를 생성하도록 구성되고, 상기 데이터는 생산 데이터와 무관한 (absent), 어플리케이션을 테스트하기 위한 장치.

청구항 9

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 네거티브 데이터를 생성하기 위한 네거티브 데이터 제조기를 더 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 10

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 엔지니어링 된 테스트 데이터를 생성하기 위한 수단을 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 11

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 상기 엔지니어링 된 테스트 데이터의 참조 (referential) 완전성 (integrity)을 결정하기 위한 완전성 확인기를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 12

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 참조 완전성 내의 오류들을 검출하도록 더 구성되는, 어플리케이션을 테스트하기 위한 장치.

청구항 13

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 데이터를 엔지니어링 된 테스트 데이터로서 출력하기 전에 상기 데이터 내의 참조 완전성의 손실을 정정하기 위한 재-참조기 (re-referncer) 를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 14

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 데이터 내의 참조 완전성의 손실을 정정하도록 더 구성되는, 어플리케이션을 테스트하기 위한 장치.

청구항 15

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 상기 엔지니어링 된 테스트 데이터를 수신하고 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 관측 (view) 하는 것 및 상기 엔지니어링 된 테스트 데이터를 프로파일링하는 것 중 적어도 하나를 가능하게 하기 위한 조사 유닛을 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 16

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 상기 엔지니어링 된 테스트 데이터를 수신하고 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 관측하는 것을 가능하게 하기 위한 데이터 조사 유닛을 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 17

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 상기 엔지니어링 된 테스트 데이터를 수신하고 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 프로파일링하는 것을 가능하게 하기 위한 프로파일러를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 18

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 프로파일링하는 것을 가능하게 하도록 더 구성된, 어플리케이션을 테스트하기 위한 장치.

청구항 19

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 관측하는 것을 가능하게 하도록 더 구성된, 어플리케이션을 테스트하기 위한 장치.

청구항 20

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 엔지니어링 된 테스트 데이터를 생성하기 위한 복수의 수단을 포함하고, 엔지니어링 된 테스트 데이터를 생성하기 위한 특정한 수단은 상기 테스트 될 어플리케이션에 관한 정보에 적어도 부분적으로 기초하여 생성되는, 어플리케이션을 테스트하기 위한 장치.

청구항 21

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 각각 상기 엔지니어링 된 테스트 데이터를 위한 기초를 형성하는 데이터를 제공하도록 구성된, 데이터 강화기, 데이터 증류기, 네거티브 데이터 제조기, 및 포지티브 데이터 제조기를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 22

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 데이터 증류기로 논리적 기능들을 제공하고 테스트 될, 상기 테스트 될 어플리케이션 내의 논리적 기능들을 식별하도록 구성된 논리 추출기 (logic extractor) 를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 23

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은, 생산 데이터의 서브세트를 획득하기 위한 기초로서 사용될 논리적 기능들을 제공하고 테스트 될, 상기 테스트 될 어플리케이션 내의 논리적 기능들을 식별하도록 더 구성되는, 어플리케이션을 테스트하기 위한 장치.

청구항 24

제 1 항에 있어서,

상기 계산 환경 관리자는, 테스트 될 상기 테스트 될 어플리케이션이 테스트 되는 계산 환경을 자동적으로 설립 (set up) 하고 해체 (take down) 하기 위한 수단을 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 25

제 1 항에 있어서,

상기 계산 환경 관리자는 환경 천이 머신을 포함하고, 상기 환경 천이 머신은 상기 엔지니어링 된 테스트 데이터의 소스를 식별하도록 구성되고, 상기 환경 천이 머신은 상기 테스트 될 어플리케이션에 의한 상기 엔지니어링 된 테스트 데이터의 처리로부터 유래한 데이터를 배치하기 위한 타겟을 식별하도록 더 구성된, 어플리케이션을 테스트하기 위한 장치.

청구항 26

제 1 항에 있어서,

상기 환경 천이 머신은 제 1 저장소 (repository) 로부터 상기 소스로 엔지니어링 된 테스트 데이터를 복사하도록 더 구성된, 어플리케이션을 테스트하기 위한 장치.

청구항 27

제 26 항에 있어서,

상기 환경 천이 머신은 상기 타겟으로부터 제 2 저장소로 엔지니어링 된 테스트 데이터를 복사하도록 더 구성된, 어플리케이션을 테스트하기 위한 장치.

청구항 28

제 1 항에 있어서,

상기 계산 환경 관리자는 환경 백업 머신 및 복원 머신을 포함하고, 상기 환경 백업 머신은 제 1 환경을 제 2 환경으로 변환하기 전에 상기 제 1 환경을 백업하도록 구성되고, 상기 복원 머신은 상기 제 2 환경을 상기 제 1 환경으로 대체하도록 구성되고, 상기 제 2 환경은 상기 테스트 될 어플리케이션의 테스트가 수행되는 환경인, 어플리케이션을 테스트하기 위한 장치.

청구항 29

제 1 항에 있어서,

상기 계산 환경 관리자는 집행기 (executioner) 를 포함하고, 상기 집행기는 상기 테스트 될 어플리케이션의 실행을 야기하도록 구성된, 어플리케이션을 테스트하기 위한 장치.

청구항 30

제 29 항에 있어서,

상기 집행기는 상기 어플리케이션의 실행을 야기할 때 자동적으로 스크립트를 실행하도록 구성된, 어플리케이션을 테스트하기 위한 장치.

청구항 31

제 1 항에 있어서,

상기 계산 환경 관리자는 환경 천이 머신, 환경 백업 머신, 복원 머신 및 집행기를 포함하고, 상기 환경 천이 머신은 상기 엔지니어링 된 테스트 데이터의 소스를 식별하도록 구성되고, 상기 환경 천이 머신은 상기 테스트 될 어플리케이션에 의한 상기 엔지니어링 된 테스트 데이터의 처리로부터 유래한 데이터를 배치하기 위한 타겟을 식별하도록 더 구성되고, 상기 환경 백업 머신은 제 1 환경을 제 2 환경으로 변환하기 전에 상기 제 1 환경을 백업하도록 구성되고, 상기 복원 머신은 상기 제 2 환경을 상기 제 1 환경으로 대체하도록 구성되고, 상기 제 2 환경은 상기 테스트 될 어플리케이션의 테스트가 수행되는 환경이고, 상기 집행기는 상기 테스트 될 어플리케이션의 실행을 야기하도록 구성되는, 어플리케이션을 테스트하기 위한 장치.

청구항 32

컴퓨팅 시스템 내에서 데이터를 처리하기 위한 방법으로서, 상기 방법은 어플리케이션들을 테스트하는 단계를 포함하고, 상기 어플리케이션들을 테스트하는 단계는,

데이터 처리 시스템의 입력 디바이스 및 포트 중 하나를 통해 테스트 될 어플리케이션을 나타내는 정보를 수신

하는 단계; 및

상기 수신한 정보를 처리하는 단계로서,

상기 정보에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터를 생성하는 단계,

상기 어플리케이션이 상기 엔지니어링 된 테스트 데이터 상에 동작할 계산 환경을 제어하는 단계, 및

상기 어플리케이션에 의해 동작되는 엔지니어링 된 테스트 데이터를 예상되는 출력과 비교하는 단계를 포함하는, 상기 처리하는 단계를 포함하고,

상기 방법은 상기 비교를 지시하는 결과를 출력하는 단계를 더 포함하는, 컴퓨팅 시스템 내에서 데이터를 처리하기 위한 방법.

청구항 33

어플리케이션들의 테스트를 관리하기 위한 컴퓨터 판독 가능한 매체에 비 일시적 형태로 저장된 소프트웨어로서, 상기 소프트웨어는 컴퓨팅 시스템으로 하여금,

테스트 될 어플리케이션에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터를 생성하는 단계;

상기 어플리케이션이 상기 엔지니어링 된 테스트 데이터 상에 동작할 계산 환경을 제어하는 단계;

상기 어플리케이션에 의해 동작된 엔지니어링 된 테스트 데이터를 예상되는 출력과 비교하는 단계; 및

상기 비교의 분석을 출력하는 단계를 포함하는, 처리 단계들을 실행하도록 하는 명령어들을 포함하는, 소프트웨어.

청구항 34

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은 현존 (existing) 데이터의 서브세트를 추출하도록 구성되고, 상기 데이터 엔지니어링 모듈은 상기 서브세트들을 증강 (augment) 시킴으로써, 증강된 데이터를 생성하도록 더 구성되고, 상기 엔지니어링 된 테스트 데이터는 상기 증강된 데이터를 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 35

제 34 항에 있어서,

상기 증강된 데이터는, 상기 서브세트의 하나 이상의 레코드들에 부가되는 하나 이상의 필드들을 포함하는, 어플리케이션을 테스트하기 위한 장치.

청구항 36

제 35 항에 있어서,

상기 데이터 엔지니어링 모듈은, 하나 이상의 제공된 규칙들에 기초하여, 부가된 상기 하나 이상의 필드들을 채우기 위한 데이터를 생성하도록 더 구성되는, 어플리케이션을 테스트하기 위한 장치.

청구항 37

제 1 항에 있어서,

상기 데이터 엔지니어링 모듈은 현존 데이터의 증류 (distillation) 에 의해 엔지니어링 된 테스트 데이터를 생성하도록 구성되고, 상기 엔지니어링 된 테스트 데이터는 상기 현존 데이터보다 높은 논리 집중 (logic

concentration) 을 가지는, 어플리케이션을 테스트하기 위한 장치.

발명의 설명

기 술 분 야

[0001] 본원은 미국 출원 번호 US 62/047,256 의 2014년 9월 8일의 우선일의 이익을 주장한다.

[0002] 본원은 품질 제어에 관한 것으로서, 특히 소프트웨어 어플리케이션들 내의 결함들이나 단점들을 식별하기 위해 사용되는 디바이스들 및 방법들에 관한 것이다.

배 경 기 술

[0003] 데이터 처리 머신은 그것을 일반적인 컴퓨터로부터 특정한 태스크를 수행하는 특수 목적의 머신으로 변환하기 위해 재구성을 요구한다. 결과적인 재구성은 그리하여 일반적인 컴퓨터가 이전에 할 수 없었던 일들을 수행할 수 있는 능력을 제공함으로써, 일반적인 컴퓨터를 향상시킨다. 이러한 재구성은 전형적으로 일반적인 컴퓨터로 하여금 특정한 전문화된 (specialized) 소프트웨어를 실행하도록 함으로써 수행된다. 이러한 전문화된 소프트웨어는 종종 "어플리케이션" 또는 "앱" 으로서 지칭된다.

[0004] 큰 프로젝트들에 대해, 테스트 될 어플리케이션은 엔지니어들의 팀에 의해 디자인되고 구현된다. 이후 이러한 어플리케이션은 품질 보증 팀으로 제공된다. 품질 보증 팀은 전형적으로 디자인 팀으로부터 분리된다. 품질 보증 팀은 이러한 어플리케이션 내의 결점들이나 단점들을 검색하는 것을 진행한다.

[0005] 어플리케이션을 테스트하기 위한 절차는 매우 어려울 수 있다. 이러한 어려움은 많은 이유로 발생한다. 하나의 이유는 품질 보증 팀은 근본적으로 네거티브, 다시 말해 결점들 또는 단점들이 테스트 되는 소프트웨어 내에 존재하지 아니한다는 것을 입증하려고 노력한다는 것이다. 일반적으로, 모든 가능한 경우들을 다루기 위해 많은 수의 테스트들을 실행하는 것은 비용 효율적이지 않다. 그러므로 분별력 있게 테스트 데이터를 선택하는 것이 필수적이다.

[0006] 어플리케이션을 테스트하기 위한 절차 내에서의 또 다른 어려움은 테스트가 수행되는 환경이 차이를 만들 수 있다는 것이다. 환경은 일반적으로 실행하는 소프트웨어, 및 어플리케이션이 동작하고자 하는 데이터 양쪽 모두를 포함한다. 다른 소프트웨어가 실행하는 것을 아는 것은 테스트 되는 어플리케이션과 그 소프트웨어 사이의 인터랙션들의 경우에 중요하다. 테스트 되는 어플리케이션의 특징들은 상기 어플리케이션으로 제공되는 데이터에 매우 종속되므로 정확한 데이터 프레젠트 (present) 를 가지는 것이 중요하다. 예를 들어, 어플리케이션은 데이터베이스로부터 특정 데이터를 요청할 수도 있다. 이러한 경우에, 어플리케이션을 테스트하는 것은 데이터베이스가 정확한 데이터를 가진다는 것을 알 것을 요구한다. 따라서, 품질 보증 팀은 일반적으로 환경을 제어하기 위한 조치를 취한다.

[0007] 어플리케이션을 테스트 함에 있어서 발생하는 또 다른 어려움은 결과들의 완전성 (integrity) 을 확립하는 것이다. 어떤 경우에, 특정한 환경 내에서 처리되는 입력 데이터의 주어진 입력 세트에 대해 결과들이 "정확" 또는 "부정확"하다고 여겨져야 하는지 아는 것은 매우 어려울 수 있다.

[0008] 테스트는 소프트웨어 개발 주기의 주요한 부분이기 때문에, 이를 수행하기 위한 보다 효율적인 방법을 제공하는 것이 유용하다.

발명의 내용

과제의 해결 수단

[0009] 일 측면에서, 본 발명은 어플리케이션들을 테스트하기 위한 장치를 특징으로 한다. 이러한 장치는, 메모리 및 상기 메모리에 동작 가능하게 커플링 된 프로세서를 가지는 데이터 처리 머신을 포함한다. 상기 데이터 처리 머신은 데이터 엔지니어링 모듈, 계산 환경 관리자, 및 결과 분석 모듈을 포함하는 데이터 구동 테스트 프레임워크를 구현하도록 구성된다. 상기 데이터 엔지니어링 모듈은, 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터를 생성하도록 구성된다. 한편, 상기 계산 환경 관리자는, 상기 어플리케이션이 상기 엔지니어링 된 테스트 데이터 상에 동작할 계산 환경을 제어하도록 구성된다. 마지막으로, 상기 결과 분석 모듈은, 상기 어플리케이션에 의해 동작되는 엔지니어링 된 테스트 데이터를 예상되는 출력과 비교하도록

구성된다.

[0010] 일부 실시예에서, 상기 데이터 엔지니어링 모듈은, 생산 (production) 데이터의 서브세트를 추출하도록 구성된다. 이러한 서브세트는, 명시된 코드 범위 (coverage) 를 달성하도록 선택된다. 상기 엔지니어링 된 테스트 데이터는, 상기 생산 데이터의 이러한 서브세트를 포함할 수도 있다.

[0011] 다른 실시예에서, 상기 데이터 엔지니어링 모듈은, 생산 데이터로부터 증류된 (distilled) 데이터를 생성하기 위한 데이터 증류기 (still) 를 포함한다.

[0012] 본 발명의 범위 내에 또한 포함되는 것은, 상기 데이터 엔지니어링 모듈은 생산 데이터의 서브세트를 추출하고, 추가 데이터와 함께 상기 서브세트들을 증강 (augment) 시킴으로써, 증강된 데이터를 생성하도록 구성되는 실시예이다. 상기 추가 데이터는 명시된 코드 범위를 달성하도록 선택되고, 엔지니어링 된 테스트 데이터는 증강된 데이터를 포함한다.

[0013] 일부 실시예에서, 상기 데이터 엔지니어링 모듈은 데이터 증류기 및 데이터 증류기로부터 증류된 데이터를 수신하고 상기 증류된 데이터를 강화 (enhance) 하기 위한 데이터 강화기 (enhancer) 를 포함한다.

[0014] 추가적인 실시예는 데이터 엔지니어링 모듈이 상기 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 데이터를 생성하도록 구성되는 것을 포함한다. 상기 생성된 데이터는 명시된 코드 범위를 달성하도록 선택되고, 상기 엔지니어링 된 테스트 데이터는 상기 생성된 데이터를 포함한다.

[0015] 다른 실시예는, 상기 데이터 엔지니어링 모듈이 포지티브 데이터를 생성하기 위한 포지티브 데이터 제조기를 더 포함하는 것, 상기 데이터 엔지니어링 모듈이 상기 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 데이터를 생성 - 상기 데이터는 생산 데이터와 무관함 (absent) - 하도록 구성되는 것, 상기 데이터 엔지니어링 모듈이 네거티브 데이터를 생성하기 위한 네거티브 데이터 제조기를 더 포함하도록 구성되는 것을 포함한다.

[0016] 일부 실시예에서, 상기 데이터 엔지니어링 모듈은, 엔지니어링 된 테스트 데이터를 생성하기 위한 수단을 포함한다.

[0017] 추가적인 실시예는 상기 데이터 엔지니어링 모듈이 상기 엔지니어링 된 테스트 데이터의 참조 (referential) 완전성 (integrity) 을 결정하기 위한 완전성 확인기를 포함하는 것뿐만 아니라, 상기 데이터 엔지니어링 모듈이 참조 완전성 내의 오류들을 검출하도록 더 구성되는 것을 포함한다.

[0018] 또한 포함되는 것은, 상기 데이터 엔지니어링 모듈이 데이터를 엔지니어링 된 테스트 데이터로서 출력하기 전에 상기 데이터 내의 참조 완전성의 손실을 정정하기 위한 재-참조기 (re-referncer) 를 포함하는 실시예, 및 상기 데이터 엔지니어링 모듈이 데이터 내의 참조 완전성의 손실을 정정하도록 더 구성되는 실시예이다.

[0019] 추가적인 실시예들은 상기 데이터 엔지니어링 모듈이 상기 엔지니어링 된 테스트 데이터를 수신하고 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 관측 (view) 하는 것 또는 상기 엔지니어링 된 테스트 데이터를 프로파일링하는 것을 가능하게 하기 위한 조사 유닛을 포함하는 것, 상기 데이터 엔지니어링 모듈이 상기 엔지니어링 된 테스트 데이터를 수신하고 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 관측하는 것을 가능하게 하기 위한 데이터 조사 유닛을 포함하는 것, 상기 데이터 엔지니어링 모듈이 상기 엔지니어링 된 테스트 데이터를 수신하고 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 프로파일링하는 것을 가능하게 하기 위한 프로파일러를 포함하는 것, 상기 데이터 엔지니어링 모듈이 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 프로파일링하는 것을 가능하게 하도록 더 구성되는 것, 그리고 상기 데이터 엔지니어링 모듈이 사용자로 하여금 상기 엔지니어링 된 테스트 데이터를 관측하는 것을 가능하게 하도록 더 구성되는 것을 포함한다.

[0020] 일부 실시예에서, 상기 데이터 엔지니어링 모듈은 엔지니어링 된 테스트 데이터를 생성하기 위한 복수의 방법을 포함한다. 이러한 실시예에서, 어떻게 엔지니어링 된 테스트 데이터를 생성할 것인가의 선택은 상기 테스트 될 어플리케이션에 관한 정보에 적어도 부분적으로 의존한다. 다른 실시예에서, 그것은 각각 상기 엔지니어링 된 테스트 데이터를 위한 기초를 형성하는 데이터를 제공하도록 구성된, 데이터 강화기, 데이터 증류기, 네거티브 데이터 제조기, 및 포지티브 데이터 제조기를 포함한다.

[0021] 또한 포함되는 것은, 상기 데이터 엔지니어링 모듈이 데이터 증류기로 논리적 기능들을 제공하고 테스트 될 어플리케이션 내의 이러한 논리적 기능들을 식별하도록 구성된 논리 추출기 (logic extractor) 를 포함하는 실시예, 그리고 상기 데이터 엔지니어링 모듈이 생산 데이터의 서브세트를 획득하기 위한 기초로서 사용될 논리적 기능들을 제공하고 테스트 될 어플리케이션 내의 이러한 논리적 기능들을 식별하도록 더 구성되는 실시예이다.

[0022] 추가적인 실시예에서, 상기 계산 환경 관리자는, 상기 어플리케이션의 테스트가 수행될 계산 환경을 자동적으로 설립 (set up) 하고 해체 (take down) 하기 위한 수단을 포함한다.

[0023] 본 발명의 실시예에 또한 포함되는 것은, 상기 계산 환경 관리자가 환경 천이 머신을 포함하는 실시예이다. 상기 환경 천이 머신은 상기 엔지니어링 된 테스트 데이터의 소스를 식별하도록 구성되고, 상기 테스트 될 어플리케이션에 의한 상기 엔지니어링 된 테스트 데이터의 처리로부터 유래한 데이터를 배치하기 위한 타겟을 식별하도록 더 구성된다.

[0024] 일부 실시예에서, 상기 환경 천이 머신은 제 1 저장소 (repository) 로부터 상기 소스로 엔지니어링 된 테스트 데이터를 복사하도록 더 구성된다. 이러한 실시예 중에는 상기 환경 천이 머신이 상기 타겟으로부터 제 2 저장소로 엔지니어링 된 테스트 데이터를 복사하도록 더 구성된 것이 포함된다.

[0025] 본 발명의 실시예들은 상기 계산 환경 관리자가 환경 백업 머신 뿐만 아니라 복원 머신을 포함한다. 이러한 실시예에서, 상기 환경 백업 머신은 제 1 환경을 제 2 환경으로 변환하기 전에 상기 제 1 환경을 백업하도록 구성되고, 상기 제 2 환경은 상기 테스트 될 어플리케이션의 테스트가 수행되는 환경이다. 상기 복원 머신은 상기 제 2 환경을 상기 제 1 환경으로 대체하도록 구성된다.

[0026] 일부 실시예에서, 상기 계산 환경 관리자는 상기 테스트 될 어플리케이션의 실행을 야기하도록 구성된 집행기 (executioner) 를 포함한다. 이러한 실시예들 중에는 상기 집행기가 상기 어플리케이션의 실행을 야기할 때 자동적으로 스크립트를 실행하도록 구성된 것이 포함된다.

[0027] 또 다른 실시예는 환경 천이 머신, 환경 백업 머신, 복원 머신 및 집행기를 가지는 계산 환경 관리자를 포함한다. 이러한 실시예에서, 상기 환경 천이 머신은 상기 엔지니어링 된 테스트 데이터의 소스를 식별하도록 구성되고, 상기 환경 천이 머신은 상기 테스트 될 어플리케이션에 의한 상기 엔지니어링 된 테스트 데이터의 처리로부터 유래한 데이터를 배치하기 위한 타겟을 식별하도록 더 구성되고, 상기 환경 백업 머신은 제 1 환경을 제 2 환경으로 변환하기 전에 상기 제 1 환경을 백업하도록 구성되고, 상기 제 2 환경은 상기 테스트 될 어플리케이션의 테스트가 수행되는 환경이다. 복원 머신은 상기 제 2 환경을 상기 제 1 환경으로 대체하도록 구성된다. 그리고, 상기 집행기는 상기 테스트 될 어플리케이션의 실행을 야기하도록 구성된다.

[0028] 또 다른 측면에서, 본 발명은 컴퓨팅 시스템 내에서 데이터를 처리하기 위한 방법을 특징으로 포함한다. 이러한 방법은 어플리케이션들을 테스트하는 단계를 포함한다. 이 경우의 어플리케이션들을 테스트하는 단계는, 데이터 처리 시스템의 입력 디바이스 및 포트 중 하나를 통해 테스트 될 어플리케이션을 나타내는 정보를 수신하는 단계 및 상기 수신한 정보를 처리하는 단계를 포함한다. 이러한 수신한 정보를 처리하는 단계는 상기 정보에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터를 생성하는 단계, 상기 어플리케이션이 상기 엔지니어링 된 테스트 데이터 상에 동작할 계산 환경을 제어하는 단계 및 상기 어플리케이션에 의해 동작되는 엔지니어링 된 테스트 데이터를 예상되는 출력과 비교하는 단계, 및 상기 비교를 지시하는 결과를 출력하는 단계를 포함한다.

[0029] 또 다른 측면에서, 본 발명은 어플리케이션들을 테스트하기 위한 컴퓨팅 시스템을 특징으로 포함한다. 이러한 컴퓨팅 시스템은 정보를 기억하기 위한 수단, 및 정보를 처리하기 위한 수단을 포함한다. 정보를 처리하기 위한 수단은 데이터 구동 테스트를 위한 수단을 포함한다. 이러한 데이터 구동 테스트를 위한 수단은 데이터 처리 시스템의 입력 디바이스 및 포트의 하나 또는 양쪽 모두를 통해 정보를 수신하기 위한 수단을 포함한다. 이러한 정보는 테스트 될 어플리케이션을 나타낸다. 데이터 구동 테스트를 위한 수단은, 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터의 콜렉션을 생성하기 위한 수단 뿐만 아니라, 어플리케이션이 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터의 콜렉션을 생성하기 위한 수단에 의해 생성된 엔지니어링 된 테스트 데이터 상에 동작할 계산 환경을 관리하기 위한 수단, 및 어플리케이션에 의해 동작된 엔지니어링 된 테스트 데이터 및 예상되는 출력을 서로 비교하기 위한 수단을 포함한다. 컴퓨팅 시스템은 결과들의 분석들을 출력하기 위한 수단을 더 포함한다.

[0030] 또 다른 측면에서, 본 발명은 어플리케이션들의 테스트를 관리하기 위한 컴퓨터 판독 가능한 매체에 비 일시적 형태로 저장된 소프트웨어를 특징으로 포함한다. 이러한 소프트웨어는 컴퓨팅 시스템으로 하여금, 특정 처리 단계들을 수행하도록 하는 명령어들을 포함한다. 이러한 처리 단계들은, 테스트 될 어플리케이션에 적어도 부분적으로 기초하여 엔지니어링 된 테스트 데이터를 생성하는 단계, 상기 어플리케이션이 상기 엔지니어링 된 테스트 데이터 상에 동작할 계산 환경을 제어하는 단계, 상기 어플리케이션에 의해 동작된 엔지니어링 된 테스트 데이터를 예상되는 출력과 비교하는 단계 및 상기 비교의 분석을 출력하는 단계를 포함한다.

[0031] 본 발명의 이러한 특징들 및 다른 특징들은 하기의 상세한 설명 및 수반하는 도면들로부터 명확해질 것이다.

도면의 간단한 설명

[0032] 도 1 은 어플리케이션 테스트 머신을 위한 데이터 구동 테스트 프레임워크의 컴포넌트들 사이의 구조적 관계들의 도시이다.

도 2 는 사용자 인터페이스로부터의 스크린을 도시한다.

도 3 은 복수의 확장된 박스들과 함께 도 2 의 스크린을 도시한다.

도 4 는 도 3 에서 명시된 입력 및 출력 데이터파일들을 사용하여 테스트 되는 그래프를 도시한다.

도 5 는 입력 데이터파일을 구성하기 위한 옵션들을 도시한다.

도 6 은 베이스라인을 구성하기 위해 정보를 구체화하기 위한 박스를 도시한다.

도 7 은 레코드-바이-레코드 (record-by-record) 비교를 위한 옵션들을 도시한다.

도 8 은 테스트가 실제로 정확하게 실행했는지에 관한 정보를 도시한다.

도 9 는 베이스라인에 대비하여 어플리케이션 테스트 결과들의 요약을 도시한다.

도 10 은 다른 확장된 박스들과 함께 도 2 의 스크린을 도시한다.

도 11 은 소스 레벨 코드 범위에 대한 예시적인 리포트를 도시한다.

도 12 는 도 1 의 데이터 구동 테스트 프레임워크에 도시된 데이터 서브세터 (subsetter) 의 컴포넌트들 사이의 구조적 관계들의 도시이다.

도 13 은 도 1 의 데이터 구동 테스트 프레임워크에 도시된 데이터 제조기 (manufacturer) 의 컴포넌트들 사이의 구조적 관계들의 도시이다.

도 14 는 도 1 의 데이터 구동 테스트 프레임워크에 도시된 데이터 증강기 (augmenter) 의 컴포넌트들 사이의 구조적 관계들의 도시이다.

도 15 는 도 1 의 데이터 구동 테스트 프레임워크에 도시된 환경 관리 머신의 컴포넌트들 사이의 구조적 관계들의 도시이다.

도 16 은 효율적인 테스트 절차의 개관 (overview) 이다.

발명을 실시하기 위한 구체적인 내용

[0033] 보다 효율적인 테스트는 좋은 데이터가 테스트를 위해 이용 가능한 것, 알려진 환경 내에서 어플리케이션의 반복 가능한 테스트들을 자동적으로 실행하기 위한 방법을 제공하는 것, 정확함 (correctness) 을 측정하거나 또는 그렇지 않으면 테스트 하의 어플리케이션의 성능을 평가하기 위해 사용될 수 있는 결과들을 수집하는 것, 및 이러한 결과들을 평가하기 위한 방법을 가지는 것에 의해 달성될 수 있다.

[0034] 도 1 은 테스트 컴퓨터 (12) 상에서 어플리케이션 (14) 의 체계적이고 효율적인 테스트를 가능하게 하기 위한, 테스트 컴퓨터 (12) 에 설치된 데이터 구동 테스트 프레임워크 (10) 를 도시한다. 여기에서 사용되는 바와 같이, "테스트 컴퓨터"는 어플리케이션 테스트 절차를 수행하기 위해 협력하는 하나 이상의 처리 시스템들을 포함하는 경향이 있다.

[0035] 도 2 는 데이터 구동 테스트 프레임워크 (10) 가 어플리케이션 (14) 의 테스트와 관련된 사용을 위해 제공하는 사용자 인터페이스의 제 1 스크린을 도시한다. 제 1 스크린은 10 개의 박스들을 가진다. 클릭 되었을 때, 도 3 에 도시된 바와 같이 이러한 박스들의 각각은 사용자에게 많은 선택들을 제공하는 추가적인 박스들을 드러내기 위해 확장한다. 도 1 및 도 2 양쪽 모두의 박스들은 어플리케이션 (14) 의 테스트 동안 일반적으로 수행되는 태스크들의 순서에 따르는 방법으로 좌로부터 우로의 컬럼들 내에서 배열된다.

[0036] 도 2 의 제 1 컬럼은 "싱글 테스트 (Single Test)" 박스, "입력 데이터세트 (Input Datasets)" 박스, 및 "출력 데이터세트 (Output Datasets)" 박스를 도시한다.

[0037] 도 3 에서의 그것의 확장된 형태에서 도시된 바와 같이, "싱글 테스트" 박스는 사용자가 특정한 테스트를 구성

하는 것, 테스트 데이터세트가 유지될 곳을 명시하는 것, 그리고 테스트 환경의 설립 (set-up) 또는 해체 (tear-down) 를 위한 커스텀 로직을 구현하거나, 또는 테스트 결과들의 분석을 수행하기 위해 사용될 임의의 그래프들, 플랜들, 또는 스크립트들을 식별하는 것을 가능하게 한다.

[0038] "입력 데이터세트" 및 "출력 데이터세트" 박스들은 사용자가 입력 및 출력 데이터세트의 위치를 명시하는 것을 가능하게 한다. 일반적으로, 출력 데이터세트는 어플리케이션 (14) 이 변경하는 것인 반면에, 입력 데이터세트는 어플리케이션 (14) 이 상기 출력 데이터세트를 어떻게 변경할지 결정하기 위해 사용하는 것이다. 예를 들어, 어플리케이션 (14) 은 복수의 자동 대리점 (dealership) 들의 각각으로부터의 수익 (revenue) 에 대한 일일 리포트들을 수신할 수 있고 축적된 수익들의 데이터베이스를 업데이트할 수 있다. 이러한 경우, 업데이트 될 데이터베이스는 "출력" 데이터세트일 수 있고 일일 수익 리포트들은 "입력" 데이터세트일 수 있다.

[0039] 도 3 에 도시된 특정한 실시예는 도 4 에 도시된 그래프를 테스트하는 것과 연관된다. 이러한 그래프는 5 개의 입력 데이터세트들과 2 개의 출력 데이터세트들을 특징으로 포함한다. 도 3 에서, 이러한 데이터세트들의 이름은 "입력 데이터세트" 및 "출력 데이터세트" 박스들 내에서 적절하게 나열된다.

[0040] 도 5 는 테스트 프레임워크의 도 3 의 "A-Customers" 데이터베이스에 대한 집중을 도시하는 것을 디스플레이하는 입력 구성 박스를 도시한다. 입력 구성 박스는 사용자가 데이터세트의 이름 및 유형을 식별하는 것을 가능하게 한다. 데이터세트 유형의 예시는 입력 파일들 및 입력 데이터베이스 테이블들을 포함한다. 입력 구성 박스는 또한 사용자가 입력 데이터세트의 상태를 명시하는 것을 가능하게 한다. 데이터세트 상태의 예시는 데이터세트가 압축되었는지 또는 아닌지 여부이다. 입력 구성 박스는 또한 사용자가 입력 데이터세트로의 경로를 명시하는 것, 그리고 데이터세트의 레코드 포맷을 지시하는 것을 가능하게 한다. 테스트 프레임워크 (10) 는 명시된 입력 및 출력 데이터세트들 각각에 대해 유사한 박스를 도시한다.

[0041] 어플리케이션이 데이터 상에 동작할 때, 그것은 전형적으로 특정 방법에 따라 그것을 변경한다. 어플리케이션 (14) 이 정확하게 데이터를 변경하는지 또는 아닌지 여부는 어플리케이션 (14) 이 정확하게 동작하는지 또는 아닌지 여부에 대한 중요한 단서를 제공한다. 그러나, 단순히 변경된 데이터를 조사하고 그것이 정확한지 또는 부정확한지 선언하는 것은 일반적으로 가능하지 않다. 일반적으로, 변경된 데이터를 정확하다고 알려진 다른 데이터와 비교하는 것이 필수적이다. 정확하다고 알려진 데이터는 "베이스라인 (baseline)"이라고 불린다.

[0042] 제 1 스크린의 제 2 컬럼은 어플리케이션 (14) 이 정확하게 그것의 기능들을 수행하였는지 여부에 대한 확인과 관련 있는 박스들을 포함한다. 이러한 제 2 컬럼은 "베이스라인 비교 (Baseline Comparison)" 박스 및 "메트릭 (Metrics)" 박스를 특징으로 포함한다.

[0043] "메트릭" 박스는 사용자가 표시되어야 하는 어플리케이션의 실행에 관한 통계들을 명시하는 것을 가능하게 하기 위한 옵션들을 제공한다. 예를 들어, 이것은 경과 시간, CPU 시간, 및 코드 범위 (coverage) 를 포함한다.

[0044] "베이스라인 비교" 박스는 사용자가 베이스라인 데이터를 식별하고 그것의 베이스라인으로서의 사용을 위한 준비 내에서 그에 대한 어떤 동작들을 수행하는 것을 가능하게 한다. 예를 들어, 그것은 베이스라인 데이터가 출력 데이터 내에 표시되지 않은 어떤 필드들을 가지는 것, 또는 베이스라인 데이터 내의 어떤 필드들이 출력 데이터 내의 대응하는 필드들에 본질적으로 (inherently) 매칭되지 않을 것이 될 수 있다. 일 예시는 도와줄 수 없으나 양쪽 경우들에서 상이한, 날짜/시간 스탬프가 될 수 있다.

[0045] 도 6 은 테스트 프레임워크의 도 3 의 "베이스라인 비교" 박스 내의 "베이스 라인 구성 (Configure Baseline...)" 옵션에 대한 집중을 도시하는 것을 디스플레이하는 베이스라인 구성 박스를 도시한다. 베이스라인 구성 박스는 사용자에게 비교의 유형을 선택할 수 있는 기회를 제공한다. 비교 유형들의 예시는 테스트 데이터세트 저장소 내의 MFS 파일 또는 시리얼 파일 사이의 비교일 수 있다. 베이스라인 구성 박스는 또한 사용자에게 베이스라인이 어디에 위치하는지, 그것이 압축되어있는지 또는 아닌지, 그것의 레코드 포맷, 및 비교에 앞서 드롭 (drop) 하기 위한 임의의 베이스라인 필드들 또는 출력 필드들을 명시할 수 있는 기회를 제공한다.

[0046] 도 3 에 도시된 바와 같이, 베이스라인과 어플리케이션 (14) 의 출력 간의 비교를 수행하기 위한 두 방법들이 존재한다. 한 방법은 레코드-바이-레코드 (record-by-record) 비교를 수행하는 것이다. 이것은 도 3 에서 "레코드-바이-레코드 비교를 구성 (Configure Record-by-record comparison)" 에 의해 지시된다. 다른 방법은 레코드-바이-레코드 비교 없이 병합된 데이터를 조사하는 것이다. 이것은 도 3 에서 "통계적 비교를 구성 (Configure statistical comparison...)" 에 의해 지시된다. 이것의 예시는 예상되는 레코드들의 개수에 대응하는 데이터세트 내의 레코드들의 개수를 결정할 수 있다.

[0047] 도 5, 6 은 도 3 의 "베이스라인 비교" 박스 내의 "레코드-바이-레코드 비교를 구성"을 클릭하는 것으로 이용

가능한 옵션들을 도시한다. 이용 가능한 옵션들은 비교될 키들을 명시하는 것, 및 비교 내에서 어떤 필드들을 포함할 것인지를 명시하는 것을 포함한다. 예를 들어, 동일한 시간은 한번 이상 발생할 수 없기 때문에 본질적으로 매칭되지 않을 날짜/시간 스탬프를 만약 필드가 포함하는 경우에 이것은 유용하다.

[0048] 제 3 컬럼은 테스트의 실제 실행을 제어하기 위한 싱글 테스트 실행 (Single-Test-Run) 박스를 포함한다. 싱글 테스트 실행 박스는 단지 베이스라인 분석을 실행할 뿐만 아니라 기록적인 결과들을 유지하기 위한 옵션들을 허용한다.

[0049] 제 4 컬럼 및 마지막 컬럼은 결과들의 분석을 위한 옵션들을 포함한다. 다양한 리포트들이 생성될 수 있다. 그러나 실제 테스트의 결과들을 조사하기에 앞서, 테스트가 실제 정확하고 동작했는지 결정하는 것이 유용하다. 특히, 모든 입력 및 출력 파일들이 정확하게 명시되었는지, 그리고 실제로 그것을 실행하고 있는 테스트 설립, 및 결과들의 분석의 단계들이 모두 성공적으로 완료되었는지를 확인하는 것이 유용하다. 이것은 제 4 컬럼 내의 "싱글 테스트 결과 (Single Test Results)" 박스 내의 "실행하기 위한 이벤트 디테일들을 관측 (View Event Details for Run)"을 선택하는 것에 의해 수행될 수 있다. 이것은 도 8 에 도시된 바와 같이 리포트를 산출할 것이다. 도 8 에 도시된 리포트에 따르면, 특정한 분석 단계를 제외한 모든 것이 잘 진행되었다. 무엇이 잘못 진행되었는지에 대한 디테일들은 리포트에 대한 추가적인 클릭에 의해 식별될 수 있다.

[0050] 테스트가 사용자가 만족할 정도로 실행되었는지의 결정 이후에, 테스트의 결과를 베이스라인 결과들에 비교하는 리포트들의 조사가 가능하다. 이러한 리포트들의 하나는, 도 9 에 도시된, 어플리케이션 (14) 의 테스트에 의해 산출된 결과들 및 베이스라인 사이의 비교의 요약이다. 이러한 리포트는 도 3 의 "베이스라인 비교 결과 (Baseline Comparison Results)" 박스 내의 "요약 관측 (View Summary)" 을 클릭하는 것에 의해 획득된다. 리포트는 차이를 가지는 레코드들의 개수 및 베이스라인 레코드들의 개수를 도시한다. 나타난 바와 같이, 도 9 의 테스트 결과는 테스트 된 어플리케이션이 많은 오류를 만들었다는 것을 제시한다.

[0051] 어플리케이션이 얼마나 많은 오류를 만들었는지와 어디에서 그들이 발생하였는지를 관찰하는 것에 부가하여, 코드 범위에 대한 리포트를 관측하는 것 또한 가능하다. 코드 범위는 그래프 레벨, 컴포넌트 레벨, 및 종류 (kind) 레벨 범위 메트릭들을 포함하는, 다양한 방법들로 표현될 수 있다. 이용 가능한 선택들은 도 3 의 "코드 범위 결과 (Code Coverage Results)" 박스를 클릭하는 것에 의해 보여질 수 있다. 이것은 박스를 도 10 에 도시된 선택들을 드러내도록 확장한다.

[0052] 도 11 은 소스 레벨 범위 메트릭들에 대한 리포트의 예시를 도시한다. 이러한 리포트는 도 10 의 "코드 범위 결과" 박스 내의 "소스 레벨 코드 범위 메트릭 관측 (View Source-Level Code Coverage Metrics)" 를 클릭하는 것에 의해 획득된다.

[0053] 도시된 데이터 구동 테스트 프레임워크 (10) 는 데이터 구동 테스트 프레임워크 (10) 의 설치에 앞서 테스트 컴퓨터 (12) 에 존재하지 않았던 기능들을 테스트 컴퓨터 (12) 에 제공한다. 이러한 방법으로, 도시된 데이터 구동 테스트 프레임워크 (10) 는 그것이 설치된 테스트 컴퓨터 (12) 의 동작에 있어서 상당한 기술적 향상을 제공한다.

[0054] 테스트 될 어플리케이션 (14) 은 소스 코드들의 편집 (compilation) 을 통해 획득한 오브젝트 코드를 포함할 수 있다. 특정 실시예에서, 이러한 소스 코드는 방향성을 가지는 비순환 (acyclic) 그래프들을 나타낸다. 다른 실시예에서, 소스 코드는 계획들을 나타낸다.

[0055] 일부 실시예에서, 소스 코드는 그래프들을 나타낸다. 이러한 그래프들의 노드들은 컴포넌트들 간의 데이터 흐름을 가능하게 하기 위한 방향성을 가지는 링크들에 의해 연결된 포트들을 가지는 처리 컴포넌트들을 정의한다. 이러한 그래프에서, 컴포넌트들은 입력 포트들 상에서 입력 데이터를 수신하는 것, 그 데이터를 처리하는 것, 및 출력 포트들에서 결과적인 출력을 제공하는 것에 의해 계산을 수행한다.

[0056] 일부 실시예에서, 소스 코드는 계획들을 나타낸다. 계획은 노드들이 태스크들을 나타내고 방향성을 가지는 링크들이 태스크들 간의 종속 관계들을 정의하여 다운스트림 태스크들이 업스트림 태스크들이 종료되기 전에 시작하지 못하는, 방향성을 가지는 비순환 그래프들이다. 일부 실시예에서, 태스크는 그래프를 실행하기 위해 사용된다.

[0057] 어플리케이션 (14) 과 관련된 컴파일 된 소스 코드는 "pset," 또는 파라미터 세트를 나타내는 정보를 포함할 수 있다. 파라미터 세트는 파라미터들의 리스트 및 이러한 파라미터들 각각에 대응하는 값들을 제공한다. 일부 실시예에서, 파라미터 세트는 그래프를 커스터마이징 하기 위한 파라미터들을 제공하기 위해 사용된다.

[0058] 어플리케이션 (14) 은 그로부터 그들이 유도된 소스 코드가 데이터 흐름 그래프들, 제어 흐름 그래프들 및 계획들을 나타내는 것에 한정되지 않는다. 실시예는 또한 어플리케이션 (14) 이 C 코드 또는 자바 코드와 같은 임의의 컴퓨터 언어로 작성된 소스 코드의 적절한 편집 또는 해석에 의해 획득되는 오브젝트 코드를 포함하는 것을 포함한다. 이러한 어플리케이션들의 실행의 추가적인 설명은, 그 내용이 본원에서 참조로서 병합되는, 2014년 8월 7일에 공개된 미국 공개특허공보 2014-0222752 의 Isman 등의 "DATA RECORDS SELECTION" 내에서 제공된다.

[0059] 어플리케이션 (14) 은 종종 그 실행이 하나 이상의 변수들의 값에 의해 트리거되는 규칙들을 시행한다. 이러한 변수들은 입력 데이터에 대응하는 입력 변수들일 수 있다. 또는 그들은 입력 데이터 내의 하나 이상의 입력 변수들에 의존하는 유도된 변수들일 수도 있다. 어플리케이션의 효율적인 테스트를 위해, 어플리케이션 (14) 내의 모든 논리 규칙의 실행을 야기하기에 충분한 테스트 데이터를 제공하여 어플리케이션 내의 완전한 코드 범위가 달성되는 것이 때때로 바람직하다. 논리 규칙이 적어도 대응하는 최소 횟수 실행되는 것을 야기하거나, 또는 반대로, 논리 규칙이 대응하는 최대 횟수 이하로 실행되도록 하는 것을 야기하는 것이 또한 바람직할 수 있다.

[0060] 효율적인 테스트에 대한 제 1 장애물은, 어플리케이션 (14) 에 의해 동작되었을 때 그에 대한 적절한 테스트 데이터를 획득하는 것은 앞서 말한 요건들을 충족할 것이라는 점이다. 여기에서 고려된 특정한 테스트 데이터는, 각각 하나 이상의 필드들로 구성된 레코드들의 시리즈로서 구성된 데이터이다.

[0061] 테스트 데이터를 습득하기 위한 하나의 방법은 생산 시스템으로부터 당겨진 (pulled) 완전한 데이터 용량 (volume) 을 사용하는 것이다. 이론적으로, 이러한 방법은 코드의 어떤 특징의 테스트를 생략할 확률이 점근적으로 0 으로 접근할 정도로 큰 데이터의 용량을 테스트하는 것에 의존한다.

[0062] 이러한 데이터 용량은 종종 매우 컸다. 그 결과, 각각의 테스트 사이클은 비합리적으로 긴 시간을 소요할 수 있었다.

[0063] 앞서 말한 장애물을 극복하기 위해, 도시된 데이터 구동 테스트 프레임워크 (10) 는 어플리케이션 (14) 의 테스트에 사용하기 위한 엔지니어링 된 테스트 데이터를 생성하는 데이터 엔지니어링 모듈 (16) 을 포함한다. 어떻게 엔지니어링 된 테스트 데이터를 생성하는지의 예시는 2013년 12월 18일에 출원된 Isman 등의 미국 가출원 61/917,727 인 "DATA GENERATION," 과 2013년 3월 14일에 출원된 Isman 등의 미국 출원 13/827,558, 미국 특허공개공보 2014/0222752 의 "DATA RECORDS SELECTION," 양쪽 모두에 개시되어 있다. 앞서 말한 출원들 양쪽의 내용은 본원에 참조로서 병합된다.

[0064] 여기에 개시된 데이터 구동 테스트 프레임워크 (10) 는 총 데이터 용량은 코드 범위가 종속하는 유일한 것이 아니라는 발견을 이용하고자 한다. 실제로, 코드 범위는 데이터 그 자체의 특성에 역시 종속한다. 특히, 코드 범위는 그 데이터의 논리 집중 또는 논리 분포에 종속한다. 실제는, 더 높은 논리 집중을 가지도록 엔지니어링 되는 테스트를 위해 실제 사용되는 데이터를 제공받아, 극단적으로 더 작은 양의 데이터를 사용하여 원하는 코드 범위를 종종 달성할 수 있다.

[0065] 여기에서 사용되는 바와 같이, 용어 "코드 범위"는 소스 코드가 테스트 절차에 의해 테스트 된 규모의 척도 (measure) 이다. 이것은 백분율로서 종종 표현되는, 제 2 값에 대한 제 1 값의 비율 - 여기서, 제 2 값은 테스트 될 코드의 총량의 양적인 척도를 나타내고, 제 1 값은 테스트 될 실제 양의 양적인 척도를 나타냄 - 로서 표현될 수 있다. 일부 경우에서, 제 1 및 제 2 변수들은 구현된 특징들에 대한 테스트 된 특징들을 나타낸다. 다른 경우에서, 제 1 및 제 2 변수들은 테스트 된 소스 코드의 라인들 및 소스 코드의 총 라인들을 나타낸다. 양적인 척도들의 정확한 속성은 본 발명의 이해를 위해 명백하게 중요하지 않다.

[0066] 데이터 구동 테스트 프레임워크 (10) 는 100% 의 코드 범위는 고사하고, 임의의 특정한 코드 범위를 달성할 것이 요구되지 않는다. 코드 범위는 엔지니어링 판단에 기초하여 사용자에 의해 설정되는 파라미터이다. 그러나 사용자가 선택한 코드 테스트 범위가 무엇이든지, 본원에 개시된 방법들 및 장치들은 그러한 범위를 달성하기 위해 요구되는 테스트 데이터의 양을 감소시킬 것이고, 생산 데이터의 전체 용량의 단순한 조정 (manipulation) 에 의해 아마 달성될 수 있는 것보다 더욱 믿을 수 있고 결정론적인 (deterministic) 방법으로 타겟 코드 테스트 범위를 달성할 것이다.

[0067] 특히, 주어진 테스트 데이터의 세트에 대해, 코드의 어떤 부분들이 행사될 (exercised) 것이다. 상이한 테스트 데이터세트들은 일반적으로 코드의 상이한 부분들을 행사할 것이다. 예를 들어, 만약 테스트 데이터가 단순히 데이터 레코드를 반복한다면, 그것은 코드의 매우 제한된 서브세트만을 행사할 것이다. 반대로, 값들의 모든 부류의 조합들을 가지는 다양한 레코드들을 포함하는 테스트 데이터는 코드의 큰 서브세트를 실행할 것이다.

[0068] 데이터 엔지니어링 모듈 (16) 은 컴포넌트 세트로부터 선택된 하나 이상의 컴포넌트들을 포함한다. 각 컴포넌트는 특정한 방법을 사용하여 엔지니어링 된 테스트 데이터를 생성한다. 어떤 방법을 사용할 것인지의 선택, 그리고 그에 따라 어떤 컴포넌트가 요구되는지는 가까이의 특정한 환경들에 종속한다.

[0069] 데이터 엔지니어링 모듈 (16) 의 컴포넌트들은 하나 이상의 데이터 서브세터 (18), 데이터 증강기 (20), 포지티브 데이터 제조기 (22), 및 네거티브 데이터 제조기 (24) 를 포함한다. 데이터 서브세터 (18) 는 현존 (existing) 데이터의 증류 (distillation) 를 통해 엔지니어링 된 테스트 데이터를 생성하여 그것의 논리 집중을 향상시킨다. 데이터 증강기 (20) 는 현존 데이터를 증강함으로써 엔지니어링 된 테스트 데이터를 생성한다. 포지티브 데이터 제조기 (22) 및 네거티브 데이터 제조기 (24) 의 양쪽 모두는 테스트의 요건들에 기초하여 엔지니어링 된 테스트 데이터를 생성한다.

[0070] 어플리케이션 (14) 내의 어떤 로직을 테스트하기 위해 요구되는 데이터의 종류들이 현존 데이터에서 나타나지 않는 경우들이 존재한다. 그러나, 이것은 그러한 로직이 테스트 되지 못한다는 것을 의미하지 않는다.

[0071] 만약 이러한 로직을 행사하기 위한 테스트 데이터에만 의존하면 로직은 절대로 테스트 되지 않을 것이다. 이는 현존 데이터의 증류의 양은 그 로직을 테스트하기 위해 사용될 수 있는 데이터를 산출하는 것을 보장하지 않을 것이기 때문이다. 이러한 환경들을 수용하기 위해, 데이터 엔지니어링 모듈 (16) 의 어떤 실시예들은 네거티브 데이터 제조기 (24) 를 포함한다.

[0072] 네거티브 데이터 제조기 (24) 는 보통은 나타나지 않는 데이터를 제공한다. 이는 그렇지 않으면 테스트 될 기회를 전혀 가지지 않을 수 있는 코드의 행사를 가능하게 함으로써 테스트의 코드 범위를 확장한다. 네거티브 데이터 제조기 (24) 는 보통 전형적인 데이터세트 내에 (또는 전형적인 데이터세트의 샘플 내에) 나타나지 않을 수 있는 데이터 - 본원에서 "네거티브 데이터"로 지칭됨 - 를 제공하기 때문에, 네거티브 데이터 제조기 (24) 는 포지티브 데이터 제조기 (22) 와 상이하다. 반대로, 포지티브 데이터 제조기 (22) 는 보통 전형적인 데이터세트 내에 (또는 전형적인 데이터세트의 샘플 내에) 나타나는 데이터 - 본원에서 "포지티브 데이터"로 지칭됨 - 를 생성한다. 네거티브 데이터의 예시는, 그 필드에 대한 문자 (character) 들의 미리 정의된 세트 내에 있지 않은 문자를 포함하는 필드 엔트리 (field entry), 또는 그 필드에 대한 값들의 미리 정의된 범위의 밖에 있는 값을 가지는 필드 엔트리, 또는 그 필드 엔트리의 하나 이상의 부분들 내의 부정확한 개수의 문자들을 포함하는 필드 엔트리와 같이, 필드의 포맷으로 부적절한 필드 엔트리들을 포함한다. 실시예는 글자, 또는 0 의 값을 가지는 출생 월을 포함하는 사회 보장 번호일 수 있다. 네거티브 데이터의 다른 실시예는 필드 포맷과 일치하지만 그럼에도 불구하고 참조 (referential) 완전성 (integrity) 을 방해하는 것을 포함한다. 실시예는 어떠한 현존 고객도 식별하지 않는 확실하게 포맷된 고객 번호일 수 있다. 이러한 네거티브 테스트 케이스들의 사용은 코드 범위를 향상시킨다. 그러나, 이러한 네거티브 데이터는 생산 데이터세트에 나타나지 않을 것이고, 그러므로 일반적으로 제조를 요구할 것이다.

[0073] 생성된 엔지니어링 된 테스트 데이터를 가지는 것의 결과로서, 어플리케이션 (14) 이 개발되고 있는 동안 그 어플리케이션 (14) 의 인터랙티브 디버깅을 용이하게 수행하는 것이 가능해진다. 이것은 실행하기 위해 많은 분 (minute) 들, 또는 심지어 시간들을 소요할 수도 있는 큰 데이터세트들의 처리보다 훨씬 생산적이다. 예를 들어, 엔지니어링 된 테스트 데이터가 국지적인 환경에서 사용될 때, 업무 규칙 환경에서 규칙들을 바꾸는 것의, 각각의 레코드에 대한, 효과를 보는 것이 가능해진다.

[0074] 데이터 서브세터 (18) 는 어플리케이션 (14) 의 개발자들이 그 어플리케이션 (14) 에 대해 만들어진 변화들의 효과를 빠르게 볼 수 있을 정도로 충분히 작은 엔지니어링 된 테스트 데이터의 세트를 산출한다. 그러나 엔지니어링 된 테스트 데이터의 세트는 단지 작은 것 그 이상이다. 그것은 또한 높은 테스트 논리 집중을 가진다. 그것의 높은 테스트 로직 집중의 결과로서, 엔지니어링 된 테스트 데이터는 전체 데이터세트들을 요구하지 않고서도 어플리케이션 (14) 내의 모든 코드들을 시행한다. 이것은 계산 자원들의 동일한 소비로 높은 코드 범위를 달성하는 것을 야기한다.

[0075] 도 12 는 데이터 서브세터 (18) 의 디테일들을 도시한다. 데이터 서브세터 (18) 는 실제 생산 데이터 (26) (또는 서브세팅을 위한 임의의 입력 데이터세트), 논리 명세 (specification) 및 제어 변수 (30) 를 수신한다. 논리 추출기 (31) 는 테스트 될 논리적 기능들을 식별하고 그들을 데이터 증류기 (32) 로 제공하며, 이들 양쪽 모두는 데이터 서브세터 (18) 의 구성 성분 (constituent) 들이다. 데이터 증류기 (32) 는 이후 생산 데이터 (26) 를 처리하여 데이터 증류물 (distillate) (33) 을 생성한다. 그것은 제어 변수 (30) 에 의해 명시된 바에 따른 추출 절차를 사용하는 논리 추출기 (31) 에 의해 명시된 논리 테스트와 연관된 부분들을 추출하는 것에 의해 그렇게 행동한다. 그리하여, 여기에서 사용되는 바와 같이, 용어 "데이터 증류"는 입력 데이터세트로부터 데이터

의 부분을 추출하기 위한 명시된 추출 절차를 사용하고, "데이터 증류물"이라 불리는 추출된 데이터를 산출하는 처리 모듈을 지칭하기 위해 사용된다.

[0076] 데이터 증류물 (33) 은 서브세팅 규칙들에 기초하여 생산 데이터 (26) 로부터 선택된다. 이러한 서브세팅 규칙들은 복수의 소스들로부터 유래할 수 있다. 하나의 예시에서, 사용자는 서브세팅 규칙들을 명시한다. 다른 예시에서, 서브세팅 규칙들은 어플리케이션의 실행으로부터의 피드백에 기초하여 만들어진다. 또 다른 예시에서, 데이터 증류물 (33) 은 실행될 어플리케이션 (14) 의 코드의 일부 또는 전부를 야기할 수 있는 데이터 레코드들을 포함한다.

[0077] 하나의 실시예로서, 생산 데이터 (26) 는 각각 필드들을 포함하고 일부 필드들이, 그들 중 일부는 나머지들에 비해 발생할 가능성이 높은 어떤 허용된 값들을 가지는 데이터 레코드들을 포함할 수 있다. 상이한 허용된 값들은 코드의 상이한 부분들을 행사한다. 그리하여, 코드들을 철저하게 테스트하기 위해, 모든 값들의 모든 조합들은 반드시 발생해야 한다. 일부 실시예에서, 엔지니어링 된 테스트 데이터는 이러한 가능성 낮은 값들을 보다 발생할 가능성이 높아지게 하여 너무 많지 않은 레코드들이 허용된 값들의 모든 조합들을 획득하기 위해 요구되지 않도록 함으로써 유도된다.

[0078] 이 경우에, 엔지니어링 된 테스트 데이터는 레코드의 값들의 확률 분포가 더 균일하게 만들어진 데이터로서 관측될 수 있다. 다시 말해, 만약 특정한 허용된 값이 생산 데이터 (26) 내에서 비교적 낮은 확률로 발생하면, 그 값은 엔지니어링 된 테스트 데이터 내에서 더 높은 확률로 발생할 것이다. 반대로, 특정한 허용된 값이 생산 데이터 (26) 내에서 비교적 높은 확률로 발생하면, 그 값은 엔지니어링 된 테스트 데이터 내에서 더 낮은 확률로 발생할 것이다. 이것은 가장 공산이 큰 이벤트들의 확률이 감소하고 가장 공산이 작은 이벤트의 확률이 증가하는 엔지니어링 된 테스트 데이터를 가지는 것의 총 효과를 가진다. 이것은 확률 값들의 확산을 감소시킨다. 확률 값들의 확산이 0인, 이것의 제한된 경우는 균일한 분포의 정의에 의한다. 확률 값들의 전체 확산의 감소는 그리하여 분포를 균일한 분포로 이끄는 경향이 있다. 동일한 시점에 더 개연성 없는 값들을 획득하는 것을 보장하기 위해 요구되는 용량이 감소하는 반면, 보다 개연성 있는 값들에 의해 야기된 리던던시 (redundancy) 들이 감소하므로, 이것은 테스트를 위한 더욱 효율적인 데이터세트를 도출하는 경향이 있다. 이러한 효율성의 정도는 엔지니어링 된 테스트 데이터의 테스트 논리 집중에 대응한다.

[0079] 많은 경우에 있어, 생산 데이터 (26) 는 데이터베이스로부터의 복수의 테이블들로 구성될 것이다. 이러한 테이블들은 제 2 테이블 내의 레코드를 가리키거나, 또는 "참조"하는 제 1 테이블 내의 포인터를 가지는 것에 의해 커플링 될 수 있다.

[0080] 포인터가 어떤 것을 가리킬 때마다, (1) 포인터가 어떤 유효한 것을 가리키는 것, 그리고 (2) 포인터가 어떤 유효한 것을 가리키지 않는 것의 두 가능성이 있다.

[0081] 제 1 가능성에서, 제 1 테이블 내의 각각의 포인터는 제 2 테이블 내의 유효한 레코드를 가리킨다. 이러한 제 1 가능성에서, 이 두 테이블들은 "참조 완전성"을 가지는 것으로 일컬어진다. 따라서, 여기에서 사용되는 바와 같이, 용어 "참조 완전성"은 데이터세트 (들) 의 어느 한 부분의 데이터세트 (들) 의 다른 부분의 값에 대한 각각의 참조가 유효한 하나 이상의 데이터세트들을 설명하기 위해 사용된다.

[0082] 상기 개시된 제 2 가능성에서, 제 1 테이블 내의 적어도 하나의 포인터는 제 2 테이블 내의 유효한 레코드를 가리키지 않는다. 이러한 제 2 가능성에서, 이 두 테이블들은 참조 완전성이 결핍하다고 일컬어진다.

[0083] 적절한 테스트를 위해, 만약 생산 데이터 (26) 가 참조 완전성을 가지면 엔지니어링 된 테스트 데이터 역시 그렇게 해야만 하는 것이 바람직하다. 그리하여, 데이터 증류기 (32) 는 참조 완전성을 유지하는 데이터 증류물 (33) 을 제공해야 한다.

[0084] 이러한 참조 완전성이 유지되었는가 여부를 결정하기 위해, 데이터 증류기 (32) 는 데이터 증류물 (33) 을 완전성 확인기 (34) 로 제공한다. 만약 완전성 확인기 (34) 가 데이터 증류물 (33) 이 참조 완전성을 가진다고 결정하면, 데이터 증류물 (33) 은 데이터 서브세터 (18) 의 출력 데이터 서브세트 (35) 로서 제공된다. 그렇지 않으면, 그것은 복구를 위해 재-참조기 (36) 로 제공되고, 그 이후 그것은 출력 데이터 서브세트 (35) 로서 제공된다.

[0085] 일부 실시예에서, 재-참조기 (36) 는 데이터 증강기 (20) 와 동일한 기능을 구현한다. 예를 들어, 하나의 데이터세트의 포인터가 다른 데이터세트 내의 레코드를 지시하지 않아서 참조 완전성의 결핍이 발생하면, 재-참조기 (36) 는 데이터 증강기 (20) 에 의해 사용되는 방법과 동일한 방법을 사용하여 적절한 레코드로 제 2 데이터세트를 증강시킬 수 있다. 재-참조기 (36) 는 그리하여 데이터 엔지니어링 모듈 (16) 의 선택적인 구성 성분으로

서 관측될 수 있다.

[0086] 특정한 실시예에서, 도시된, 데이터 서브세터 (18) 는 또한 출력 데이터 서브세트 (35) 를 프로파일 및/또는 관측하는 것을 가능하게 하는 데이터 조사 유닛 (37) 을 포함한다. 그러나 다른 실시예에서, 데이터 조사 유닛 (37) 은 존재하지 않는다.

[0087] 데이터 조사 유닛 (37) 을 가지는 실시예에 속하는 것은, 데이터 조사 유닛 (37) 이 관측기 (viewer) 인 것, 그리고 데이터 조사 유닛 (37) 이 프로파일러 (profiler) 인 것이다. 또한, 데이터 조사 유닛 (37) 을 포함하는 실시예에 포함되는 것은, 데이터 조사 유닛 (37) 이 사용자가 무엇을 수행하기 원하는지에 기초하여 관측 및 프로파일링 양쪽 모두가 가능한 구조를 가지는 것이다.

[0088] 여기서 사용되는 바와 같이, 데이터 서브세트를 "프로파일링" 하는 것은 예를 들어 메타데이터를 획득하는 것, 또는 그 서브세트에 관한 데이터를 종합 (aggregate) 하는 것을 포함할 수 있고, 프로파일링의 결과는 "프로파일"로 일컬어진다. 데이터를 종합하는 것은 레코드들의 개수, 이러한 레코드들의 값들의 범위, 그리고, 확률 분포의 n 번째 모멘트 (n 은 양의 정수) 와 같은, 데이터 내의 값들의 통계적 또는 확률론적 서술 (description) 들과 같은 특징들을 포함한다.

[0089] 예를 들어 새로운 시스템을 개발할 때, 때때로 증류하기 위해 이용 가능한 생산 데이터가 존재하지 않는다. 다른 경우에, 생산 데이터는 획득하기에 매우 어려울 수도 있다. 이러한 환경들을 수용하기 위해, 데이터 엔지니어링 모듈 (16) 의 포지티브 데이터 제조기 (22) 를 활성화한다.

[0090] 도 13 을 참조하면, 포지티브 데이터 제조기 (22) 는 논리 명세 (28), 제어 변수 (30) 및 키-관계 정보 (38) 를 수신한다. 논리 추출기 (31) 는 테스트 될 논리 기능들을 식별하고 이들을 데이터 생성기 (40) 로 제공한다. 데이터 생성기 (40) 는 제어 변수 (30) 에 의해 명시된 바에 따른 추출 절차를 이용하여 적절한 테스트 데이터를 생성한다. 어떻게 데이터를 생성하는지에 대한 실시예는 2013년 12월 18일에 출원된 Isman 등의 미국 가출원 61/917,727 인 "DATA GENERATION,", 그리고 2014년 8월 7일에 공개된, Isman 등의 미국 공개특허공보 2014/0222752 인 "DATA RECORDS SELECTION," 에 개시된다.

[0091] 가급적이면, 결과적인 제조된 테스트 데이터 (39) 는 적절한 테스트를 위한 참조 완전성을 가진다. 따라서, 제조된 테스트 데이터 (39) 는 참조 완전성이 설립되었는가 여부를 결정하기 위해 완전성 확인기 (34) 로 제공된다. 만약 완전성 확인기 (34) 가 제조된 데이터가 참조 완전성을 가진다고 결정하면, 제조된 테스트 데이터 (39) 는 포지티브 데이터 제조기 출력 (41) 으로서 제공된다. 만약 제조된 테스트 데이터가 참조 완전성을 가지지 않으면, 제조된 테스트 데이터 (39) 는 복구를 위해 재-참조기 (36) 로 제공되고 그 이후 포지티브 데이터 제조기 (22) 의 출력으로서 제공된다.

[0092] 일부 실시예에서, 포지티브 데이터 제조기 (22) 는 데이터 구동 테스트 프레임워크 (10) 내에서 제조된 테스트 데이터 (39) 를 관측 또는 프로파일하는 것을 가능하게 하는 데이터 조사 유닛 (37) 을 또한 포함한다. 다른 실시예에서, 데이터 조사 유닛은 존재하지 않는다.

[0093] 일부 경우에, 생산 데이터 (26) 는 존재하지만 요구되는 형태에 전적으로 해당하지 않는다. 이러한 경우에, 데이터 엔지니어링 모듈 (16) 의 데이터 증강기 (20) 를 활성화 시키는 것에 의해 생산 데이터를 증강시키는 것이 유용하다.

[0094] 데이터 증강기 (20) 는, 예를 들어, 현존 생산 데이터 (26) 에 하나 이상의 필드들을 부가하고 공급된 규칙들에 기초하여 이러한 필드들을 채우기 위한 데이터를 생성하기 위해 사용될 수 있다.

[0095] 도 14 는 데이터 증강기 (20) 의 디테일들을 도시한다. 데이터 증강기 (20) 는 실제 생산 데이터 (26) (또는 증강될 임의의 입력 데이터세트), 논리 명세 (28) 및 제어 변수 (30) 를 수신한다. 논리 추출기 (31) 는 테스트 될 논리 기능들을 식별하고 그들을 데이터 증류기 (32) 및 데이터 수정기 (modifier, 48) 양쪽 모두로 제공한다. 데이터 증류기 (32) 는 생산 데이터 (26) 를 처리하여, 제어 변수 (30) 에 의해 명시된 바에 따른 추출 절차를 이용하여 논리 추출기 (31) 에 의해 명시된 로직 테스트와 연관된 부분들을 추출하도록 한다. 로직 추출기 (31) 에 의해 제공된 정보에 기초하여, 데이터 수정기 (48) 는 적절한 필드들을 추가하고 그 필드들로 적합한 값들을 입력하며, 그리하여 증강된 데이터 (49) 를 생성한다.

[0096] 가급적이면, 데이터 수정기 (48) 에 의해 제공된 증강된 데이터 (49) 는 적절한 테스트를 위한 참조 완전성을 가진다. 따라서, 데이터 수정기 (48) 에 의해 제공된 증강된 데이터 (49) 는 참조 완전성이 유지되었는가 여부를 결정하기 위해 완전성 확인기 (34) 로 제공된다. 만약 완전성 확인기 (34) 가 증강된 데이터 (49) 가 참조

완전성을 가진다고 결정하면, 증강된 데이터 (49) 는 데이터 증강기 (20) 의 증강된 데이터 출력 (51) 으로서 제공된다. 그렇지 않으면, 증강된 데이터 (49) 는 복구를 위해 재-참조기 (36) 로 제공되고, 그 이후 데이터 증강기 (20) 의 증강된 데이터 출력 (51) 로서 제공된다.

[0097] 일부 실시예에서, 데이터 증강기 (20) 는 데이터 구동 테스트 프레임워크 (10) 내에서 증강된 데이터 출력 (51) 을 관측 및 프로파일하는 것을 가능하게 하는 데이터 조사 유닛 (37) 을 또한 포함한다. 다른 실시예에서, 데이터 증강기 (20) 는 데이터 조사 유닛을 가지지 않는다.

[0098] 일부 경우에, 생산 데이터 내에 보통 나타날 수 있는 임의의 데이터에 의해 행사되지 않을 수 있는 코드 세그먼트들을 행사하기 원할 수 있다. 이를 수행하기 위해, 데이터 엔지니어링 모듈은 네거티브 데이터 제조기 (24) 를 포함하고, 그것의 기능은 이러한 네거티브 테스트 경우들을 생성하는 것이다.

[0099] 효율적인 테스트에 대한 제 2 장애물은, 테스트 환경을 설립 (set up), 제어, 그리고 해체 (tear down) 할 필요로부터 대두된다.

[0100] 일반적으로, 테스트는 테스트 스위트 (suite) 내에서 복수의 테스트들을 실행하는 것 및 많은 외부 데이터세트들과 함께 상호작용하는 하나 이상의 그래프들 및 계획들에 대해서도 그렇게 하는 것을 포함한다. 이러한 데이터세트들은 파일들, 테이블들, 큐 (queue) 들, 멀티-파일들 및 웹 서비스들로부터 유래할 수 있다. 어플리케이션 (14) 이 테스트 스위트들을 실행하는 것을 야기하는 태스크를 달성하기 위해, 데이터 구동 테스트 프레임워크 (10) 는 계산 환경 관리자 (44) 를 제공한다.

[0101] 계산 환경 관리자 (44) 는 알려진 환경 내에서 알려진 입력들과 함께 제어된 방법으로 어플리케이션 (14) 을 실행하는 태스크를 수행한다. 이것은 테스트 될 특정한 어플리케이션 (14) 을 명시함에 있어서 유연성을 제공한다. 계산 환경 관리자 (44) 는 어플리케이션 (14) 에 의해 처리될 입력 데이터에 대응하는 종합 (aggregate) 데이터, 데이터 플래그들, 출력 디렉토리, 그리고 설립, 해체 및 리포팅에 대한 커스터마이징 가능한 로직을 포함하는 저장소 폴더를 유지한다.

[0102] 계산 환경 관리자 (44) 는 파일들 또는 테이블들로서 데이터세트들을 자동적으로 설립한다. 이러한 데이터세트들은 데이터의 소스들, 다시 말해 어플리케이션 (14) 이 동작할 데이터, 그리고 타겟들, 다시 말해 어플리케이션 (14) 에 의한 처리의 결과들이 궁극적으로 배치될 곳을 포함한다. 환경 관리자 (44) 는 이후 소스 및 타겟을 정확한 초기 상태로 자동적으로 설정하고, 적절한 테스트 스위트를 이용하여 어플리케이션 (14) 를 실행하고, 결과들을 타겟 내에 배치하고, 그리고 환경을 그것의 프리셋 (pre-set) 조건으로 되돌린다. 일부 경우에, 환경 관리자 (44) 는 이전의 환경을 백업하고 테스트가 완료된 이후에 이를 복구한다. 환경의 자동화된 설립 및 해체는 최소화된 수동적 노동과 함께 반복적인 테스트를 가능하게 한다.

[0103] 컴퓨터 시스템은 계속하여 증가하는 관념 (abstraction) 의 네스팅 된 (nested) 레이어들의 세트로서 관측될 수 있다. 각각의 레이어는 상위 레벨의 관념에서 레이어에 의해 활용될 수 있는 논리적 구조들을 생성한다. 이들은 메모리 상태들 및 환경 변수들의 값들을 포함한다.

[0104] 어플리케이션이 실행하였을 때, 그것은 이러한 레이어들 상에서의 실행으로서 관측될 수 있다. 하위 레이어들에 의해 생성된 논리적 구조들의 세트는 어플리케이션이 실행하는 환경으로서 관측될 수 있다. 어플리케이션의 적절한 테스트를 위해, 물리적 구조의 적절한 테스트가 종종 일정한 물리적 환경을 유지하는 것에 의존하는 것과 동일한 방법으로 동일한 환경을 유지하는 것이 바람직하다.

[0105] 도 15 를 참조하면, 일 실시예에서, 계산 환경 관리자 (44) 는 두 환경 천이들 - 설립 단계 동안의 하나 및 해체 단계 동안의 다른 것 - 을 야기하는 환경 천이 머신 (46) 을 포함한다.

[0106] 환경 천이 머신 (46) 은 입력 명세 (53) 및 출력 명세 (50) 를 수신한다. 입력 명세 (53) 는 입력 테스트 데이터가 그로부터 유래할 소스 (52) 를 식별한다. 이러한 입력은 파일들, 멀티-파일들, 큐들, 웹 서비스들, 또는 이들의 임의의 조합일 수 있다. 출력 명세 (50) 는 테스트의 출력이 배치될 것으로 추정되는 타겟 (54) 을 식별한다. 환경 천이 머신 (46) 은 입력, 출력, 및 임의의 환경 변수들의 초기 상태들에 관한 정보를 포함하는 초기화 신호 (56) 를 또한 수신한다. 최종적으로, 환경 천이 머신 (46) 은 테스트의 시작을 지시하는 테스트 신호 (58) 를 수신한다.

[0107] 일부 실시예에서, 설립 단계 동안에, 환경 천이 머신 (46) 은 제 1 데이터 저장소로부터 소스 (52) 로 테스트 데이터 및/또는 베이스라인 데이터를 복사하고, 여기서 그것은 실제 테스트 절차 동안에 저장된다. 테스트 절차가 완료된 이후에, 해체 단계가 시작한다. 이러한 해체 단계 동안에, 환경 천이 머신 (46) 은 타겟 (54) 으로부

터 테스트 데이터를 삭제한다.

[0108] 테스트 신호 (58) 를 수신하면, 환경 천이 머신 (46) 은 환경의 백업 (62) 을 생성하기 위해 환경 백업 머신 (60) 과 통신한다. 이는 입력 소스 스위치 (64) 가 적절한 소스 (52) 를 가리키는 것을 야기하는 것, 그리고 출력 소스 스위치 (66) 가 적절한 타겟 (54) 을 가리키는 것을 야기하는 것에 의해 후속된다.

[0109] 이러한 태스크들이 완료되면, 환경 천이 머신 (46) 은 실행기 (68) 가 어플리케이션 (14) 이 하나 이상의 테스트들 (80) 을 포함하는 테스트 스위트 (79) 를 실행하는 것을 야기하도록 시그널링한다. 일부 구현에서, 테스트 스위트의 실행은 하나 이상의 스크립트들의 자동화된 실행을 포함한다. 실행의 완료에 이어, 실행기 (68) 는 환경 복원 머신 (70) 을 시그널링하고, 이는 이후 백업 (62) 을 회수하고 환경을 그것의 초기 상태로 복원한다.

[0110] 실행의 과정에서, 어플리케이션 (14) 는 하나 이상의 규칙들을 구현한다. 일부 실시예에서, 규칙은 적어도 상태 (condition) 표현과 실행 표현을 포함하는 명세에 의해 명시된다. 상태 표현이 "참"으로서 평가되면 어플리케이션 (14) 은 실행 표현을 평가하는 것을 진행한다. 그러나 상태 표현이 "참"으로서 평가되는가 또는 아닌가 여부는 데이터 내의 하나 이상의 변수들의 값에 의존할 수도 있다. 이러한 변수들은 입력 데이터에 대응하는 입력 변수들일 수 있다. 또는 그들은 하나 이상의 입력 변수들에 종속하는 유도된 변수들일 수 있다. 어플리케이션 (14) 이 특정한 테스트 시행 동안 규칙을 실행하는가 아닌가의 여부는 그리하여 결국 테스트 데이터의 선택이 "참"으로 평가될 규칙에 대응하는 조건 표현을 야기할 변수들을 가지는가 여부에 의존한다.

[0111] 일부 실시예에서, 어플리케이션 (14) 은 트리거 되는 모든 규칙들을 실행한다. 다른 실시예에서, 어플리케이션 (14) 은 트리거 되는 모든 규칙들 보다 더 적게 실행한다. 규칙들은 그 내용이 본원에서 참조로서 병합되는, 2007년 4월 10일에 출원된, 미국 특허 제 8,069,129 호의 컬럼 5 의 61 줄 및 컬럼 6 의 11 줄 사이에 보다 구체적으로 개시되어 있다.

[0112] 실행기 (68) 가 테스트 스위트 (79) 를 완료하면, 결과 분석 모듈 (72) 은 테스트 결과들의 분석을 인계받아 시작한다. 결과 분석 모듈 (72) 의 기능들에 속하는 것은 정확한 결과들의 이러한 알려진 세트들을 생성하는 것과 테스트 되는 어플리케이션 (14) 이 정확한 답변들에서 결국 도착하는지를 확인하는 절차를 자동화하는 것이다.

[0113] 일부 경우에서, 테스트 되는 어플리케이션의 더 오래된 버전이 존재한다. 이러한 테스트 되는 어플리케이션의 오래된 버전은 전형적으로 현재 사용되고 있는 버전이다. 보통 말하는 바와 같이, 그것은 출력의 진실성 (veracity) 을 설립하기 위한 최적의 표준 (gold-standard) 으로서 여겨질 수 있다. 따라서, 테스트 되는 어플리케이션에 의해 대체되는 경향이 있는, 이러한 어플리케이션의 오래된 버전은, "최적 표준 버전 (gold-standard version)"으로서 지칭될 수 있다.

[0114] 테스트 되고 있는 어플리케이션의 버전이 동일한 환경을 이용하여 동일한 데이터 상에서 실행되었을 때 최적 표준 버전에 의해 획득된 것과 일치하는 결과들을 제공하지 않는다면, 테스트 되고 있는 어플리케이션의 버전은 부정확한 결과들을 출력한다는 추론이 만들어질 수 있다.

[0115] 어플리케이션 (14) 의 테스트 내에서 발현하는 하나의 단계는 어플리케이션 (14) 이 실제로 데이터를 정확하게 처리했는가 여부를 결정하는 것이다. 이 단계를 실행하기 위해, 어플리케이션 (14) 의 기능적 명세에 의해 정의되는, 데이터세트 상의 동작의 예상되는 결과와, 실행기 (68) 에 의해 획득되는 바와 같은, 동일한 데이터세트 상의 동작의 측정된 결과 사이의 어떤 대응 관계를 확립하기 위한 방안이 존재해야 한다. 다시 말해, 정확한 답변들의 베이스라인 (74) 을 획득하는 것이 필요하다. 이러한 베이스라인 (74) 이 이용 가능하면, 결과 분석 모듈 (72) 은 그들을 베이스라인 (74) 과 비교하는 것에 의해 결과들 (78) 을 확인한다.

[0116] 베이스라인 (74) 을 획득하는 방법들은 그것이 대체하는 어떤 것으로부터 얼마나 어플리케이션 (14) 이 상이한지 여부에 부분적으로 의존한다. 일반적으로, 차이가 크면 클수록, 베이스라인을 생성하기 위한 어려움은 더욱 커진다.

[0117] 관념 (abstract) 레벨에서, 주어진 데이터세트 X 및 환경 E에 대해, 어플리케이션 f의 버전 n은 출력 Y = fn (X,E) 를 생성할 것이다. 문제는 만약 Y 가 정확하다면 어떻게 결정하는가이다.

[0118] 일반적으로, 3 개의 가능성들이 존재한다.

[0119] 제 1 가능성은 (X,E) 상에 동작할 수 있는, 상이한 버전의 어플리케이션, 다시 말해 버전 m 이 존재하는 것이다. 만약 버전 m 이 신뢰 가능한 것으로 여겨지면, 만약 fn (X,E) = fm (X,E) 인지를 묻는 것에 의해 결과 Y 의 진실성을 확립한다.

[0120] 제 2 가능성은 완전히 신뢰 가능한 것으로 여겨지지 않는, 또 다른 버전의 어플리케이션, 다시 말해 버전 $\mathbb{m}$ 이 존재하는 것이다. 이 경우에, 만약 fn (Z,E) = fm (Z,E) 인지 - 여기서, Z ⊂ X 이고 fm (X,E) 는 Z 에 대해서 신뢰 가능한 것으로, 하지만 Z^C 에 대해 신뢰 가능하지 않은 것으로 여겨지고, Z^C 는 Z 의 여집합 - 를 반드시 물어야 한다. fn (Z^C, E) 의 진실성을 확립하기 위해, 전형적으로 정확한 결과들을 수동적으로 결정해야 한다.

[0121] 제 3 가능성은 신뢰 가능한 것으로 알려진 어플리케이션의 버전이 존재하지 않는 것이다. 이것은 단순히 제 2 가능성의 악화된 (degenerate) 경우이고, Z = . 이 경우에, 정확한 결과들을 결정하기 위한 절차는 수동으로 수행된다.

[0122] 베이스라인 (74) 을 획득하기 위한 하나의 방법은 테스트 하의 어플리케이션 (14) 이 현존 어플리케이션 (14) 을 근본적으로 동일한 기능으로 대체하려고 하는 경우에 유용하다. 이는 상기 정의된 제 1 가능성에 대응한다. 이 경우, 베이스라인 (74) 은 어플리케이션의 최적 표준 버전에 의해 생성되는 결과들로부터 유래할 수 있다.

[0123] 일부 경우에, 테스트 하의 어플리케이션 (14) 은 현존 어플리케이션으로의 향상을 나타낸다. 향상은 테스트 하의 어플리케이션 (14) 이 상이한 결과를 산출할 것으로 예상되고, 실제로 그렇게 하려고 하는 것이다. 상기의 제 2 가능성에 대응하는 이러한 상황은 예를 들어 최적 표준 버전이 부정확한 답변들을 야기한 버그를 가졌고 테스트 하의 어플리케이션 (14) 이 그 버그를 고치려고 하는 경우 발생할 수 있다.

[0124] 이러한 경우들에 대해, 결과 분석 모듈 (72) 은 어떤 필드들이 변경되었는가 및/또는 출력 내의 레코드들의 개수가 변경되었는가를 리포트한다. 결과 분석 모듈 (72) 은 임의의 미스매치들을 보고하여 일부 필드들이 그들이 의도하지 않았을 때 부주의로 변경된 경우 즉시 인식할 수 있도록 한다. 변경될 것으로 예상되었던 필드들에 대해서, 정확한 답변들을 결정하고 그들이 베이스라인 (74) 으로 입력되도록 하기 위해 인간의 개입이 요구된다.

[0125] 다른 경우에, 테스트 하의 어플리케이션 (14) 은 완전히 새로운 시스템이다. 이는 상기의 제 3 가능성에 대응한다. 그 결과, 베이스라인 (74) 의 생성을 위한 기초로서 사용될 수 있는 출력 데이터는 존재하지 않는다.

[0126] 이 경우, 베이스라인 (74) 은 현존 생산 데이터 (26) 와 함께 시작하는 것과 그 생산 데이터 (26) 의 서브세트를 위한 정확한 결과들을 (예들 들어, 수동으로) 입력하는 것에 의해 형성된다. 이는 테스트 될 어플리케이션 (14) 의 기저 (underlying) 논리를 관찰하는 것, 그리고 그 논리에 기초하여 어플리케이션 (14) 을 통과하는 다양한 논리에 의해 가장 많은 영향을 받을 것 같은 소스 데이터 내의 그러한 필드들을 식별하는 것에 의해 달성된다. 이들은 데이터의 서브세트를 선택할 때 채택되어야 하는 필드들이다.

[0127] 일부 경우에, 어떤 단순한 테스트는 베이스라인 (74) 을 조사할 필요 없이 자동적으로 수행될 수 있다. 예를 들어, 만약 어플리케이션 (14) 이 각각의 입력의 레코드들에 대해 하나의 출력의 레코드를 생성한다고 알려지면, 어플리케이션 (14) 는 알려진 관계 수 (cardinality) 의 생산 데이터 (26) 상에 동작하게 될 수 있고, 이 경우 출력 데이터의 관계 수는 어플리케이션 (14) 의 기능에 관한 일부 정보를 제공할 수 있을 것이다. 특히, 생산 데이터 (26) 의 개별적인 관계수들과 어플리케이션 (14) 을 이용하여 생산 데이터 (26) 상에 작동함으로써 산출된 그것 사이의 0이 아닌 차이가 존재하는 범위에 대해, 결과 분석 모듈 (72) 은 어플리케이션 (14) 의 구현 내의 결함 (flaw) 의 가능성을 자동적으로 시그널링 할 수 있다.

[0128] 예를 들어, 어떤 경우에는, 어플리케이션 (14) 은 상이한 관계 수들 사이에 관계가 존재하는 이러한 상이한 관계 수들의 여러 구성 성분들을 포함하는 출력을 생성하려고 한다. 하나의 예시에서, 어플리케이션 (14) 은 소스 (52) 내의 입력 상에 작동하고 타겟 (54) 내에 두 개의 분리된 테이블들을 생성한다. 이러한 두 테이블들의 관계 수들 사이의 관계 (relationship) 가 존재하는 범위에 대해서, 결과 분석 모듈 (72) 은 이러한 차이점을 자동적으로 검출하고 어플리케이션 (14) 의 구현 내의 결함을 지시하는 정보를 출력한다.

[0129] 다른 실시예에서, 소스 (52) 내의 입력 테이블은 N 개의 레코드들을 가질 수도 있다. 만약 타겟 (54) 내의 출력 테이블이 N 개의 레코드들을 또한 가져야 한다고 알려진 경우에, 출력 테이블 내의 레코드들의 개수를 확인하는 것은 얼마나 소프트웨어가 잘 작동할 것인지 확인하기 위한 좋은 방법이다. 예를 들어, 입력 내에 단지 N 개의 레코드들만이 존재할 때 출력 내에 N+1 개의 레코드들이 존재한다는 것을 관측하면, 이것은 오류를 제안할 것이다.

[0130] 앞서 살핀 실시예의 일반화인 다른 실시예에서, 어플리케이션은 어떤 결정론적 방법으로 레코드들의 개수를 변경하는 것으로 알려진다. 그리하여, 일반적으로, 만약 N-레코드 입력 테이블에 대한 레코드들의 출력 개수가 어떤 알려진 함수 f 에 대해 f (N) 이면, 어플리케이션 내의 오류를 식별하기 위한 하나의 방법은 입력 테이블이

N 개의 레코드들을 가질 때 출력 테이블이 실제로 f (N) 레코드들을 가지는지 여부를 관찰하는 것이다.

[0131] 실행 이후, 어플리케이션 (14) 의 실행을 지시하는 정보, 그리고 특히 테스트 데이터가 그것에 제공한 어플리케이션 (14) 의 상호작용에 관한 정보를 제공하는 리포트를 제공하는 것이 유용하다. 이러한 정보의 예시들은 어플리케이션 (14) 이 실행한 또는 실행하지 않은 규칙들, 어플리케이션 (14) 내에서 각각의 규칙이 실행된 횟수, 또는 어플리케이션 (14) 과 테스트 데이터 사이의 상호작용을 설명할 수 있는 임의의 다른 정보를 포함할 수 있다.

[0132] 리포트에 기초하여, 사용자가 추가적인 테스트 데이터를 식별하는 것이 가능하다. 이러한 추가 테스트 데이터는 예를 들어, 임의의 실행되지 않은 규칙들을 실행되도록 했을 수 있는 데이터, 또는 특정한 논리 규칙이 명시된 횟수만큼 실행되도록 했을 수 있는 데이터, 또는 다른 바람직한 실행 결과를 야기했을 수 있는 데이터가 될 수 있다. 사용자는 이후 새로운 서브세팅 규칙들을 만들어내고 이러한 추가적인 서브세팅 규칙들에 따라 데이터 레코드들의 업데이트 된 서브세트의 선택을 야기한다. 데이터 레코드들의 업데이트 된 서브세브는 이전의 실행되지 않은 규칙들의 일부 또는 전부의 실행을 야기하기에 충분한 데이터 레코드들, 명시된 횟수만큼 규칙들의 일부 또는 전부의 실행을 야기하기에 충분한 데이터 레코드들, 또는 다른 다른 바람직한 실행 결과를 야기하기에 충분한 데이터 레코드들를 포함할 수도 있다.

[0133] 결과 분석 모듈 (72) 에 의해 제공될 수 있는 정보의 유형에 속하는 것은, 테스트 데이터가 코드를 실행한 범위에 대한 리포트이다. 이 리포트는 테스트 된 코드의 라인들의 백분율과 같은 종합 스코어뿐만 아니라, 테스트되지 않은 코드의 라인들과 같은 더욱 자세한 정보들을 포함한다. 이 정보는 사용자가 테스트 된 코드의 백분율 및 테스트에서 생략된 코드의 중요성의 양쪽 모두의 관점에서 테스트가 적절했는가를 결정하는 것을 가능하게 한다.

[0134] 도 16 은 여기에 개시된 컴포넌트들을 사용하는 효율적인 테스트 절차의 전체적인 요약을 제시한다. 테스트 절차는 일반적으로 데이터 관련 단계들 (82) 및 어플리케이션 관련 단계들 (84) 로 나누어진다.

[0135] 데이터 관련 단계들 (82) 은 임의의 현존 생산 데이터 상에 프로파일을 실행하는 것을 포함한다. 이는 도 16 내의 "생산 데이터 프로파일"의 텍스트에 의해 식별되는 단계 (86) 로서 식별된다.

[0136] 다음의 데이터 관련 단계는 그 프로파일로부터 생산 데이터에 관한 어떤 종합 데이터를 획득하는 것이다. 이 단계는 "메타데이터 취득"의 텍스트에 의해 식별되는 단계 (88) 로서 도 16 에서 식별된다. "메타데이터"는 병합 데이터를 나타내는 것으로 이해된다. 이러한 병합 데이터의 예시들은 키들의 리스트, 필드 관계 수, 및 값들의 범위들을 포함하지만 이에 한정되지는 않는다.

[0137] 이러한 메타데이터, 또는 "종합 데이터"는 "참조용 온전한 서브세트 생성"의 텍스트에 의해 식별되는 도 16 의 단계 (90) 에서 식별되는 바와 같이, 데이터의 참조용 (referentially) 온전한 (intact) 서브세트의 생성을 위해 사용된다.

[0138] 일부 구현들은 네거티브 테스트 데이터를 생성하고 포함하는 것에 의해 참조용 온전한 데이터 서브세트를 증강하는 것을 포함한다. 이는 "네거티브 데이터 생성"의 텍스트에 의해 식별되는 단계 (92) 에 의해 도 16 내에서 지시된다.

[0139] 다른 구현들은 합성의 (synthetic) 데이터의 제조를 통해 참조용 온전한 데이터 서브세트를 증강하는 것을 포함한다. 이는 "신규 데이터 제조"의 텍스트에 의해 식별되는 단계 (94) 에 의해 도 16 내에서 지시된다.

[0140] 어플리케이션 관련 단계들 (84) 은 어플리케이션의 구축 (build) 또는 어떤 방식으로 그것을 강화하거나 고치는 것에 의해 현존 어플리케이션을 수정하는 것을 포함한다. 어플리케이션 구축의 단계는 도 16 에서 "APP 구축"의 텍스트에 의해 식별되고 단계 (96) 으로서 도시된다. 어떤 방식으로 그것을 강화하거나 고치는 것에 의해 현존 어플리케이션을 수정하는 단계는 도 16 에서 "APP 수정"의 텍스트에 의해 식별되고 단계 (98) 로서 도시된다. 도 16 을 통한 약어 "APP"은 어플리케이션 (14) 를 지칭하는 것으로 이해된다.

[0141] 어플리케이션 관련 단계들 (84) 은 또한, 어플리케이션의 계산 모듈들 및 어플리케이션에 의해 엑세스되거나 생성되는 데이터세트들이 서로 어떻게 의존하는가를 나타내는 의존성 (dependency) 분석과 함께 저장소로의 어플리케이션 (14) 의 확인의 단계를 포함한다. 이는 도 16 에서 "APP 체크인, 의존성 분석"의 텍스트로 레이블링되고 단계 (100) 으로서 도시된다.

[0142] "엔지니어링 된 데이터에 APP 실행"으로 레이블링되는 단계 (102) 로 도 16 내에서 지시되는 바와 같이, 어플리

케이션은 이후 엔지니어링 된 테스트 데이터 상에 작동하게 된다.

[0143] 도 16 에서 "코드 범위 리포트"의 텍스트로 레이블링되는 단계 (104) 에서 도시되는 바와 같이, 코드 범위를 결정하기 위해 결과들이 조사된다.

[0144] 이러한 범위 리포트들에 기초하여, 데이터 구동 테스트 프레임워크 (10) 는 더 나은 코드 범위를 제공하기 위해 테스트 데이터에 가해질 수 있는 수정들의 제안들을 제공한다. 이는 도 16 에서 "코드 범위를 향상시키기 위한 방법 제안"의 텍스트로 레이블링되는 단계 (106) 에서 도시된다.

[0145] 단계 (106) 의 결과는 추가 데이터의 생성 또는 현존 데이터로부터 데이터의 서브세트가 추출되는 방법의 변화들에 의해 데이터 엔지니어링 절차의 수정을 선택적으로 초래한다. 이는 도 16 에서 "데이터 엔지니어링 수정"으로 레이블링되고 단계 (108) 로서 식별된다.

[0146] 추가적으로, 출력 데이터의 완전성은 베이스라인 (74) 과 그것을 비교하는 것에 의해 평가되고, 도 16 에서 "APP 을 위한 정확한 결과 결정"으로 레이블링되고 단계 (110) 으로서 도시되는 단계이다.

[0147] 도 16 의 "어플리케이션 수정"의 텍스트에 의해 마크되는 단계 (98) 에 의해 도시되는 바와 같이, 결과들이 상이한 범위에 대해, 어플리케이션 (14) 은 차이점을 제거하기 위해 수정된다. 도 16 의 "결과를 예상되는 결과와 비교"의 텍스트로 레이블링되고 참조 번호 "112"에 의해 식별되는 단계에서 차이점이 존재하는가에 대한 결정이 수행된다.

[0148] 일부 실시예에서, 데이터 증류기 (32) 는 하나 이상의 서브세팅 규칙들에 따라 생산 데이터 (26) 를 증류한다. 서브세팅 규칙은 데이터 증류기 (32) 가 데이터 레코드들의 큰 세트로부터 선택될 데이터 레코드들의 서브세트를 식별하도록 하는 규칙이다. 결과적인 데이터 증류물 (33) 은 그리하여 원본 데이터보다 더 적게 용량을 차지하고, 더 높은 테스트 논리 집중을 가진다. 어플리케이션 (14) 이 데이터 증류물 (33) 상에 동작할 때, 더 낮은 용량의 데이터로 더 높은 코드 범위가 달성될 수 있기 때문에, 이는 궁극적으로 더 효율적인 테스트를 이끈다.

[0149] 데이터 증류기 (32) 가 의존하는 서브세팅 규칙들은 내부적으로 데이터 엔지니어링 모듈 (16) 내로부터, 데이터 구동 테스트 프레임워크 (10) 내의 다른 곳으로부터, 또는 외부 소스로부터 유래할 수 있다.

[0150] 일 실시예에서, 서브세팅 규칙들은 데이터 레코드들을 프로파일링하고 결과적인 프로파일의 분석에 기초하여 서브세팅 규칙들을 만들어내기 위해 논리 명세 (28) 를 사용하는, 논리 추출기 (31) 에 의해 제공된다. 이러한 서브세팅 규칙들은 데이터 증류물 (33) 을 생성하기 위해 그들을 사용하는 데이터 증류기 (32) 로 제공된다.

[0151] 다른 실시예에서, 서브세팅 규칙들은 특정한 테스트 데이터 상에서 어플리케이션 (14) 이 실행된 결과를 포함하는 정보에 의존하는 결과 분석 모듈 (72) 로부터 유래한다. 데이터 서브세터 (18) 는 이후, 예를 들어, 결과 분석 모듈 (72) 로부터의 리포트에 기초하는 이러한 결과들의 분석에 기초하여 서브세팅 규칙들을 만들어낸다. 이러한 규칙들은 궁극적으로 데이터 증류물 (33) 을 생성하기 위해 데이터 증류기 (32) 에 의해 실행된다.

[0152] 또 다른 실시예에서, 서브세팅 규칙들을 만들어내는 대신, 데이터 서브세터 (18) 외부 소스로부터 그들을 수신한다. 일부 경우들에서, 데이터 서브세터 (18) 는 테스트 컴퓨터 (12) 에 실제로 앉아 사용자 인터페이스를 통해 수동적으로 그들을 명시하고 있는 사용자로부터 직접 서브세팅 규칙들을 수신한다. 다른 경우들에서, 데이터 서브세터 (18) 는 테스트 컴퓨터 (12) 가 하드 디스크와 같은 비-일시적 컴퓨터 판독 가능한 저장 매체로부터 그들을 판독하도록 하는 것에 의해, 또는 테스트 컴퓨터 (12) 가 인터넷과 같은 광역 네트워크를 포함하는 네트워크와 같은, 비-일시적 컴퓨터 엑세스 가능한 전송 매체를 통해 그들을 수신하도록 하는 것에 의해 서브세팅 규칙들을 획득할 수 있다.

[0153] 외부로부터 수신하였던지 내부적으로 생성하였던지, 서브세팅 규칙은 원자로 되거나 (atomic) 또는 분자로 된다 (molecular). 원자로 된 서브세팅 규칙은 추가적인 서브세팅 규칙들로 분해될 수 없다. 분자로 된 서브세팅 규칙들은 2 이상의 원자로 된 또는 분자로 된 서브세팅 규칙들의 조합으로 구성된다. 전형적으로, 불 방식의 (Boolean) 연산자들이 분자로 된 서브세팅 규칙들을 형성하기 위해 원자로 된 서브세팅 규칙들을 연결한다.

[0154] 서브세팅 규칙은 또한 결정론적이거나 확률론적 (stochastic) 이다. 결정론적 서브세팅 규칙의 예시는 특정한 기준에 매칭하는 모든 레코드들의 선택을 야기하는 규칙이다. 확률론적 서브세팅 규칙의 예시는 특정한 기준에 매칭하는 모든 레코드들 중에서 이러한 레코드들의 둘이 랜덤으로 선택되는 것을 명시하는 규칙이다.

[0155] 일부 실시예에서, 서브세팅 규칙은 하나 이상의 타겟 데이터 필드들을 지정하고 타겟 데이터 필드들에 대한 각각의 분명한 값 또는 값 분류 (classification) 가 데이터 증류물 (33) 내에 포함된다는 것을 명시한다. 이러한

실시예를 구현하기 위해, 데이터 증류기 (32) 는 데이터 레코드들 내의 타겟 데이터 필드들에 대한 각각의 분명한 값을 식별하고 서브세팅 규칙을 만족하는 이러한 데이터 레코드들만을 가지는 데이터 증류물 (33) 을 생성한다.

[0156] 예를 들어, 50 개의 상태들의 각각에 대한 분명한 값을 가지는 "상태" 데이터 필드, 그리고 2 개의 분명한 값을 가지는 "성별" 데이터 필드는 타겟 데이터 필드들로서 식별될 수 있다. 이 경우, 데이터 증류기 (32) 는 데이터 증류물 (33) 을 위한 데이터 레코드들을 선택하여 "상태"에 대한 50 개의 값들 중 각각 및 "성별"에 대한 2 개의 값들 중 각각은 데이터 증류물 (33) 내의 적어도 하나의 데이터 레코드 내에 포함된다.

[0157] 일부 실시예에서, 데이터 서브세터 (18) 는 데이터 레코드들의 동일한 세트 이내 또는 데이터 레코드들의 상이한 세트들 간의 데이터 레코드들 중의 관계의 유형을 명시하는 서브세팅 규칙을 구현한다. 이러한 실시예에서, 데이터 증류기 (32) 는 서브세트를 위해 선택된 다른 데이터 레코드들과 함께 그들의 관계에 기초하여 데이터 레코드들을 선택한다. 예를 들어, 데이터 증류기 (32) 는 데이터 증류물 (33) 내의 포함을 위해, 고객 식별자 데이터 필드에 대해 공통의 값을 공유하는 데이터 레코드들을 선택할 수도 있다.

[0158] 데이터 서브세터 (18) 는 또한 필터링에 의존하는 서브세팅 규칙을 구현할 수 있다. 이러한 경우, 데이터 증류기 (32) 는 데이터 증류물 (33) 내에, 어떤 타겟 필드들 내에 특정한 값들을 가지는 레코드들을 포함한다. 예를 들어, 데이터 증류기 (32) 는 "상태"의 각각의 값이 적어도 한번 표현되도록 레코드들을 선택할 수도 있다. 또는, 데이터 증류기 (32) 는 "인구" 필드의 값을 고려하고 데이터 레코드들을 선택하여 "상태" 값을 가지는 레코드들의 개수가 그 상태와 관련된 "인구"의 값에 의존하도록 함으로써 분배 스킴 (apportioning scheme) 을 적용할 수도 있다.

[0159] 일부 실시예에서, 데이터 분석가 또는 어플리케이션 개발자와 같은 사용자는 서브세팅 규칙들을 제공한다. 예를 들어, 사용자는 타겟 필드들을 식별하거나 또는 데이터 레코드들 간의 관계들을 명시하고 이러한 명세를 데이터 서브세터 (18) 로 제공한다.

[0160] 다른 실시예에서, 데이터 서브세터 (18) 는 데이터 레코드들을 프로파일링하고 적절한 데이터 서브세팅 규칙들을 식별하거나 만들어내기 위해 상기 프로파일의 분석을 수행한다. 프로파일링을 수행하기 위해, 데이터 서브세터 (18) 는 연관된 데이터 레코드들에 액세스하고 그들의 어떤 특징들을 분석하여 데이터 레코드들의 프로파일을 생성한다. 이러한 특징들은 단일 데이터세트의 개별 데이터 레코드들, 데이터 레코드들의 세트 내의 데이터 필드들 간의 관계들, 및 데이터 레코드들의 상이한 세트들을 가로지르는 데이터 필드들 간의 관계들 중 하나 이상을 포함한다.

[0161] 데이터 레코드들의 세트의 프로파일은 데이터 레코드들의 세트 내의 데이터의 요약이다. 이러한 요약은 필드-바이-필드 (field-by-field) 기초에 제공될 수 있다. 프로파일은 데이터 레코드들의 세트 내의 데이터를 특징지을 수 있는 정보를 포함한다. 이러한 정보의 예시는 데이터 레코드들 내의 하나 이상의 데이터 필드들의 관계 수, 하나 이상의 데이터 필드들 내의 값들의 분류 (classification), 개별 데이터 레코드들 내의 데이터 필드들 간의 관계들, 및 데이터 레코드들 간의 관계들을 포함한다. 데이터 레코드들의 세트의 프로파일은 또한 "의사필드 (pseudofield)" 를 특징짓는 정보를 포함할 수 있다. 의사필드는 연관된 데이터 레코드들 내의 하나 이상의 데이터 필드들로부터 취득된 값들의 조정 (manipulation) 에 의해 결정된 값들과 함께 이주된 합성된 데이터 필드이다.

[0162] 데이터 레코드들의 생성된 프로파일에 기초하여, 데이터 증류기 (32) 는 어플리케이션 (14) 을 위한 좋은 코드 범위를 달성하는 데이터 레코드들의 서브세트의 선택과 연관된 데이터 레코드들의 특징들을 식별한다. 예를 들어, 데이터 레코드들의 프로파일에 기초하여, 데이터 증류기 (32) 는 어플리케이션의 유도된 변수들 및 입력 변수들과 연관될 것 같은 데이터 필드들의 조합들 또는 하나 이상의 데이터 필드들을 식별할 수 있다. 일부 경우에서, 서브세팅 규칙들은 또한 사용자로부터 또는 컴퓨터 저장 매체로부터 수신된 입력에 기초하여, 및/또는 예를 들어 결과 분석 모듈 (72) 로부터 수신된 입력에 기초하는, 어플리케이션 (14) 의 실행의 결과들에 기초하여 만들어질 수 있다.

[0163] 데이터 서브세터 (18) 는 상이한 분석적인 방법들에 기초하여 서브세팅 규칙들을 명시할 수 있다. 일부 실시예에서, 데이터 서브세터 (18) 는 개별 데이터 레코드들 내의 데이터 필드들의 분석에 기초하여 서브세팅 규칙을 명시한다. 일 실시예에서, 이는 어떤 데이터 필드들이 어플리케이션 (14) 내의 변수들에 연관될 것 같은가 결정하는 것을 포함한다. 다른 실시예에서, 데이터 서브세터 (18) 는 필드의 허용된 값들의 개수에 기초하여 타겟 데이터 필드를 식별한다. 예를 들어, "성별" 데이터 필드는 오직 2 개의 허용된 값들을 가지고 타겟 데이터 필

드로서 식별될 개연성이 있다. 다른 한편, "전화번호" 데이터 필드는 타겟 데이터 필드로서 식별될 개연성이 없다.

[0164] 또 다른 실시예에서, 데이터 서브세터 (18) 는 타겟 데이터 필드로서, 하나 이상의 데이터 필드들 내의 데이터의 조정으로부터 유래한 데이터와 함께 이주된 의사필드를 식별한다. 예를 들어, "수입" 데이터 필드 내의 데이터는 카테고리들 (예를 들어, 높음, 중간, 또는 낮음) 로 분류될 수 있고, "수입" 데이터 필드의 분류들과 함께 이주된 의사필드는 타겟 데이터 필드로서 식별될 수 있다.

[0165] 다른 실시예에서, 데이터 서브세터 (18) 는 타겟 데이터 필드 및 프로파일 내에서 지시된 바에 따른 동일한 레코드 내의 하나 이상의 다른 데이터 필드들 사이의 관계들에 기초하여 타겟 데이터 필드를 식별한다. 예를 들어, 프로파일은 데이터 필드 "상태" 및 "우편 번호"가 독립적이지 않다고 지시할 수 있다. 이러한 의존성에 기초하여, 데이터 서브세터 (18) 는 가능한 타겟 데이터 필드로서 이러한 데이터 필드들 중 어느 하나만을 고려할 수 있다.

[0166] 데이터 서브세터 (18) 는 또한 프로파일 내에서 지시되는 바에 따라 데이터 레코드들의 상이한 세트들을 가로지르는, 및/또는 데이터 레코드들의 세트 이내의 상이한 데이터 레코드들 간의 관계들의 분석에 기초하여 하나 이상의 서브세팅 규칙들을 명시할 수 있다. 예를 들어, 프로파일은 데이터 필드의 공통 값을 통해 링크될 수 있는 데이터 레코드들을 지시할 수 있다. 값을 링크하는 것의 예시는 고객 ID 데이터 필드의 값이 될 수 있다.

[0167] 데이터 서브세터 (18) 가 데이터 레코드들의 서브세트를 선택하면, 그리고 데이터 조사 유닛 (37) 이 그들의 유효성을 확인하면, 데이터 엔지니어링 모듈 (16) 은 그들을 계산 환경 관리자 (44) 로 제공하고, 이는 궁극적으로 그들이 테스트 되는 어플리케이션 (14) 에 의해 동작되도록 준비한다. 데이터 엔지니어링 모듈 (16) 은 데이터 증류물 (33) 을 포함하는 데이터 레코드들 또는 이러한 데이터 레코드들을 지시하는 데이터를 제공한다. 예를 들어, 데이터 엔지니어링 모듈 (16) 은 계산 환경 관리자 (44) 로, 데이터 증류물 (33) 을 포함하는 데이터 레코드들에 대한 식별자들 또는 이러한 데이터 레코드들에 대한 어드레스를 제공할 수 있다. 데이터 엔지니어링 모듈 (16) 은 또한 계산 환경 관리자 (44) 로 데이터 레코드들의 선택된 서브세트를 포함하는 파일을 제공할 수 있다.

[0168] 실행 이후, 결과 분석 모듈 (72) 은 데이터 증류물 (33) 상에 어플리케이션 (14) 을 실행한 결과를 지시하는 데이터를 포함하는 범위 분석 리포트를 생성한다. 일부 구현에서, 결과 분석 모듈 (72) 은 실행되었거나 실행되지 않은, 그로부터 어플리케이션 (14) 이 컴파일 된 소스 코드의 부분들을 식별하는 정보, 또는 그로부터 어플리케이션 (14) 이 컴파일 된 소스 코드의 각각의 부분이 얼마나 많이 실행되었는가를 식별하는 정보를 포함하는 범위 분석 리포트를 생성한다. 어떤 구현에서, 결과 분석 모듈 (72) 은 실행되었거나 실행되지 않은 어플리케이션 (14) 의 규칙들을 식별하는 정보, 및 어플리케이션 (14) 이 각각의 규칙을 실행한 횟수를 식별하는 정보를 포함하는 범위 분석 리포트를 생성한다. 다른 구현에서, 결과 분석 모듈 (72) 은 실행하거나 실행하지 않은, 그로부터 어플리케이션 (14) 이 컴파일 된 소스 코드의 부분들 뿐만 아니라 그로부터 어플리케이션 (14) 이 컴파일된 소스 코드의 선택된 부분들이 실행된 횟수를 식별하는 정보를 포함하는 범위 분석 리포트를 생성한다. 다른 구현에서, 결과 분석 모듈 (72) 은 그로부터 어플리케이션 (14) 이 컴파일 된 소스 코드의 특정한 부분들을 실행하기 위한 시도와 연관하여 발현된 오류들을 식별하는 정보를 포함하는 범위 분석 리포트를 생성한다. 또 다른 구현에서, 결과 분석 모듈 (72) 은 어플리케이션 (14) 이 특정한 규칙들을 실행하기 위해 시도할 때 발현된 오류들을 식별하기 위한 정보뿐만 아니라, 실행되었을 때 오류들을 초래한 이러한 규칙들의 식별을 포함하는 범위 분석 리포트를 생성한다.

[0169] 일부 구현에서, 결과 분석 모듈 (72) 은 실행되거나 실행되지 않은 규칙들을 직접 식별하는 범위 분석 리포트를 생성한다. 다른 구현에서, 결과 분석 모듈 (72) 은 각각의 논리 규칙이 실행된 횟수, 실행 동안의 어플리케이션의 각각의 변수의 값, 또는 다른 정보와 같은, 어플리케이션 (14) 의 실행에 관한 추가적인 정보를 포함하는 범위 분석 리포트를 생성한다.

[0170] 다른 구현에서, 실행되지 않은 어플리케이션 내의 각각의 논리 규칙에 대해, 결과 분석 모듈 (72) 은 그 논리 규칙에 관련한 어플리케이션 (14) 의 하나 이상의 변수들을 식별한다. 일부 구현에서, 결과 분석 모듈 (72) 은 어플리케이션 (14) 을 통하는 데이터의 흐름을 지시하는 데이터와 같은, 리포트 내에 포함된 데이터에 기초하여, 또는 어플리케이션과 관련하여 미리 로딩된 정보에 기초하여 변수들을 또한 식별한다. 일부 경우에, 결과 분석 모듈 (72) 은 논리 규칙을 실행하도록 하였을 수도 있는 각각의 변수들에 대한 값들의 범위 또는 값을 또한 식별한다. 식별되면, 데이터 엔지니어링 모듈 (16) 은 데이터 레코드들의 업데이트 된 서브세트의 후속하는 선택 내의 추가적인 서브세팅 규칙들을 명시하기 위해 변수들에 대응하는 값들의 범위들 또는 값들과 입력

데이터 필드들을 사용한다.

[0171] 예를 들어, 만약 식별된 변수가 데이터 레코드들의 데이터 필드들의 하나에 직접 대응하는 어플리케이션의 입력 변수이면, 데이터 엔지니어링 모듈 (16) 은 대응하는 데이터 필드 및 데이터 필드에 대한 값들의 범위 또는 값을 식별한다.

[0172] 예를 들어, 입력 변수가 어떤 임계값보다 클 때 어플리케이션 (14) 내의 논리 규칙이 실행하면, 데이터 엔지니어링 모듈 (16) 은 임의의 제조되거나 증류된 데이터는 반드시 입력 변수가 임계값보다 큰 값을 가지는 적어도 하나의 데이터 레코드를 포함해야만 한다고 결정한다. 이러한 정보에 기초하여, 데이터 엔지니어링 모듈 (16) 은 추가적인 서브세팅 규칙을 명시함으로써 어플리케이션 (14) 으로 제공되는 후속 데이터 레코드들이, 그 규칙에 대한 입력 변수가 임계값을 초과할 때만 실행하는 논리 규칙의 실행을 야기하기에 충분한 데이터를 포함하도록 한다.

[0173] 다른 실시예에서, 식별된 변수는 데이터 레코드들의 데이터 필드들의 하나에 직접 대응하지 않는다. 이러한 변수는 "유도된 변수"로서 지칭된다. 유도된 변수의 경우에, 데이터 엔지니어링 모듈 (16) 은 어플리케이션 (14) 의 논리에 걸친 유도된 변수의 기원를 추적하기 위해 데이터-계통을 분석한다. 이러한 데이터-계통 분석은 식별된 변수가 유도되는 입력 변수들 또는 특정한 입력 변수를 식별하는 것을 가능하게 한다. 데이터 엔지니어링 모듈 (16) 은 대응하는 데이터 필드 또는 데이터 필드들과 그 데이터 필드에 대한 값들 또는 값들의 범위를 식별한다.

[0174] 예를 들어, 만약 유도된 변수의 값이 특정한 값과 동일할 때 어플리케이션 (14) 내의 논리 규칙이 실행하면, 데이터 엔지니어링 모듈 (16) 은 유도된 값이 3 개의 입력 변수들의 논리적 조합으로부터 유도되었다고 결정하기 위한 데이터 계통 분석을 위한 명령어들을 시행한다. 유도된 변수의 논리적 유도를 따름으로써, 데이터 엔지니어링 모듈 (16) 은 특정한 유도된 변수를 달성하기 위해 이러한 3 개의 입력 변수들의 요구되는 값들을 결정한다.

[0175] 유도된 변수의 바람직한 값을 산출하기 위해 요구되는 값들의 결정은 데이터 서브세터 (18) 로 제공되고, 이는 추가적인 서브세팅 규칙을 명시하여 데이터 증류물 (33) 이 유도된 변수가 바람직한 값을 획득하고 그러므로 연관된 논리 규칙의 실행을 트리거하도록 하기에 충분한 데이터를 포함하도록 한다.

[0176] 일부 실시예에서, 범위 분석의 결과들은 또한 사용자에게 제공된다. 이에 응답하여, 사용자는 데이터 서브세터 (18) 로 추가적인 서브세팅 규칙들을 제공하거나 이전에 제공된 서브세팅 규칙들을 수정할 수 있다.

[0177] 어떤 논리 규칙들은 매우 드물게 트리거되고, 심지어 데이터 레코드들의 완전한 세트도 단지 우연히, 어플리케이션 (14) 이 그 논리 규칙을 구현하는 코드를 실행하도록 하기에 충분한 데이터를 포함할 개연성이 낮다. 완전한 데이터세트 내의 이러한 결핍을 식별하기 위해, 어플리케이션 (14) 은 입력으로서의 전체 데이터 레코드들을 이용하여 한 번 이상 실행될 수도 있다. 결과적인 리포트는 입력에 대해 선택되는 데이터 레코드들의 서브세트의 고려 없이 커버 되지 못하는 규칙들을 식별한다. 이러한 결핍을 다루기 위해, 데이터 구동 테스트 프레임워크 (10) 는 포지티브 데이터 제조기 (22) 및/또는 네거티브 데이터 제조기 (24) 를 이용하여 요구되는 데이터를 제조한다.

[0178] 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 필터링에 의한 데이터 서브세팅을 수행한다. 필터링은 포지티브 또는 네거티브 일 수 있다. 포지티브 필터링에서, 비어 있는 세트로 시작하여 특정 조건을 만족하는 데이터 레코드들만을 추가한다. 네거티브 필터링에서, 꽉 찬 데이터세트로 시작하여, 특정 조건을 만족하는 데이터 레코드들을 삭제함으로써 그것을 줄여나간다.

[0179] 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 타겟 데이터 필드들의 식별, 이러한 각각의 필드의 가능한 값들의 결정, 그리고 데이터 레코드들을 선택하여 각각의 타겟 데이터 필드에 대해 각각의 허용된 값이 적어도 한 번 나타나거나, 또는 명시된 횟수만큼 나타나도록 하는 것에 의해 데이터 서브세팅을 수행한다.

[0180] 또 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 데이터 분류에 의해 데이터 서브세팅을 수행한다. 이는 타겟 데이터 필드들을 식별하는 방법과 유사하지만, 실제 타겟 값들을 값들의 범위들로 대체한다. 그리하여, 만약 타겟 데이터 필드가 위험 평가 (risk assessment) 내에서 사용되어야 하는 콜레스테롤 레벨들을 나타내면, 범위들을 이용하여 낮음, 중간, 및 높은 수입을 나타내는 빈 (bin) 들을 정의할 수 있다. 이 경우, 데이터 레코드들은 각각의 빈, 또는 분류가 미리 정의된 레코드들의 개수를 가지도록 선택되었을 수 있다.

[0181] 추가적인 실시예에서, 데이터 엔지니어링 모듈 (16) 은 값들의 조합들에 의존함으로써 데이터 서브세팅을 수행

한다. 이는 2 개의 허용된 값들 (예를 들어, 성별) 을 가지는 제 1 필드와 12 개의 허용된 값들 (예를 들어, 탄생 월) 을 가지는 제 2 필드의 두 타겟 데이터 필드들을 고려하는 것에 의해 이해될 수 있다. 만약 각각의 가능한 값이 적어도 한번 표시되었을 것을 보장하기만을 원하였다면, 이러한 요건은 12 개의 레코드들만으로 충족될 수 있었다. 그러나 이러한 두 필드들의 가능한 모든 조합들을 가지는 것을 원할 수도 있음을 상상할 수 있다. 이 경우, 적어도 24 개의 레코드들이 선택되었어야 한다.

[0182] 데이터 서브세터 (14) 에 의해 구현될 수 있는 추가적인 방법들뿐만 아니라, 상기 방법들의 추가적인 디테일들은 본원에서 이미 참조로서 병합된 "DATA RECORDS SELECTION,"이라는 제목의 특허 공개공보에서 찾아볼 수 있다.

[0183] 참조로서 병합된 출원 "DATA GENERATION," 에서 개시된 원리들에 따른 동작을 위해 데이터 엔지니어링 모듈 (16) 은 포지티브 데이터 제조기 (22), 네거티브 데이터 제조기 (24), 및 데이터 증강기 (20) 를 사용한다.

[0184] 데이터 엔지니어링 모듈 (16) 은 사용자가 명시할 수 있는 특정한 유형의 데이터를 생성한다. 예시적인 데이터 유형은 문자열, 십진법의 정수, 날짜, 및 시간을 포함한다. 데이터 엔지니어링 모듈 (16) 은 제조된 십진법의 또는 정수의 데이터에 대한 허용된 값들의 범위, 제조된 문자열 데이터에 대한 평균 문자열 길이, 제조된 데이터 내에서 사용될 수 있는 값들 또는 글자 (character) 들의 세트, 및 다른 특성 (characteristic) 들과 같이, 제조된 데이터에 대해 한계들을 부과한다. 데이터 엔지니어링 모듈 (16) 은 현존 소스 레코드들의 하나 이상의 필드들 내의 값들의 수정, 레코드들 내의 새로운 필드들의 생성 및 이주에 의한 소스 레코드들의 증강, 또는 완전히 새로운 레코드들의 생성에 의해 데이터를 제조할 수 있다. 일부 실시예에서, 사용자는 사용자 인터페이스를 통해 구성 가능한 옵션들을 명시한다.

[0185] 데이터 엔지니어링 모듈 (16) 은 포지티브 데이터 제조기 (22) 를 이용하여 어플리케이션 (14) 에 의한 처리를 위한 데이터를 제조한다. 그것은 또한 데이터 증강기 (20) 를 이용하여 생산 데이터 (26) 와 같은 현존 데이터를 수정하거나 증강할 수 있다. 예를 들어, 데이터 증강기 (20) 는 생산 데이터 (26) 로부터 취득한 하나 이상의 필드들에 대한 값들을 수정할 수 있고, 또는 하나 이상의 신규 필드들을 생성하고 이주시키고 그들을 생산 데이터 (26) 내의 현존 데이터 레코드들로 추가할 수 있다. 포지티브 데이터 제조기 (22) 를 이용하여, 데이터 엔지니어링 모듈 (16) 은 또한 완전히 새로운 데이터 레코드들을 제조할 수 있다. 일부 실시예에서, 이러한 새로운 레코드들의 포맷은 생산 데이터 (26) 에 기초하는 반면에, 다른 실시예들에서, 사용자와 같은 외부 에이전트가 서브세팅 규칙들을 명시하는 것과 관련하여 상기 개시된 동일한 방법들을 이용하여 포맷을 명시할 것이다.

[0186] 데이터 엔지니어링 모듈 (16) 은 타겟에서 저장될 데이터를 제조한다. 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 생산 데이터 (26) 에 기초하여 데이터를 제조한다. 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 스크래치 (scratch) 로부터 데이터를 제조한다. 여기서 사용되는 바와 같이, "스크래치로부터" 제조하는 것은 명시된 특성들에 따라, 하지만 현존 데이터에 기초하지 않고 제조하는 것을 의미한다.

[0187] 생산 데이터는 파일, 데이터베이스, 파라미터 세트, 또는 데이터의 다른 소스일 수 있다. 생산 데이터 (26) 는 각각 하나 이상의 데이터의 필드들을 가지는 하나 이상의 레코드들을 포함할 수 있다. 예를 들어, 생산 데이터 (26) 는 소매점의 고객들을 위한 고객 레코드들을 저장하는 고객 데이터베이스일 수 있다. 이러한 데이터베이스 내의 각각의 레코드는 개별적인 고객을 나타낸다. 각각의 레코드는 복수의 필드들을 가질 수 있다. 생산 데이터 (26) 는 필드들의 개수, 각 필드 내의 데이터의 유형, 및 값들의 허용된 범위, 최대 허용된 값, 또는 허용된 문자들의 리스트와 같은 각 필드 내의 데이터의 특성들과 같은 레코드들의 포맷을 명시하는 레코드 포맷을 가질 수 있다. 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 스크래치로부터 데이터를 생성한다. 이러한 경우에, 데이터 소스는 제공되지 않는다.

[0188] 데이터 엔지니어링 모듈 (16) 은 데이터베이스, 파일 또는 다른 데이터 구조 내에 저장될 수 있는 구성 데이터에 기초하여 데이터를 제조한다. 구성 데이터는 사용될 데이터 생성 방법 (approach), 콘텐츠 생성 모드, 제조될 데이터의 데이터 유형, 제조될 데이터에 대한 콘텐츠 기준, 및 제조될 데이터에 대한 또 다른 구성 정보를 명시할 수 있다.

[0189] 일부 경우에서, 사용자는 테스트 컴퓨터 (12) 상에서 이용 가능한 사용자 인터페이스를 통하여, 데이터 엔지니어링 모듈 (16) 이 데이터를 제조하기 위해 사용하는 구성 데이터의 전부 또는 일부를 명시한다. 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 구성 데이터의 전부 또는 일부를 결정한다. 이 경우에, 데이터 엔지니어링 모듈 (16) 은 생산 데이터의 분석에 기초하여, 또는 타겟의 바람직한 속성들에 대한 정보에 기초하여 그렇게 한다.

[0190] 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 구성 데이터에 따라 생산 데이터 (26) 내의 현존 소스 레코드들의 하나 이상의 필드들에 대한 값들을 수정하고 수정된 레코드들을 타겟에 저장하는 것에 의해 타겟에 대한 데이터를 제조하기 위해 데이터 증강기 (20) 를 사용한다. 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 주어진 필드에 대한 값들의 전부를 수정하기 위해 데이터 증강기 (20) 를 사용한다. 예를 들어, 각각의 레코드에 대한 주어진 필드로 값이 할당되어 레코드들 전체에 걸친 주어진 필드 내의 값들의 분포가 구성 데이터에 의해 지시되는 바에 따른 타겟 분포와 매치할 수 있다. 사용자 또는 구성 데이터는 이러한 타겟 분포를 명시하기 위한 정보를 명시, 또는 제공할 수 있다.

[0191] 일부 경우에, 데이터 엔지니어링 모듈 (16) 은 주어진 필드에 대한 값들 전부보다 더 적게 수정한다. 이러한 경우에 속하는 것은 데이터 엔지니어링 모듈 (16) 이 구성 데이터에 의해 지시되는 바에 따른 명시된 기준을 충족하지 않는 값들만을 수정하는 것이다. 이러한 경우의 예시는 데이터 엔지니어링 모듈 (16) 이 그 필드에 대해 허용된 값들의 특정한 범위를 벗어나는 주어진 필드에 대한 임의의 값들을 수정하는 것이다.

[0192] 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 구성 데이터에 따라 하나 이상의 새로운 필드들과 함께 생산 데이터 (26) 의 현존 소스 레코드들을 증강하기 위해 데이터 증강기 (20) 를 사용하는 것과 이러한 증강된 레코드들을 타겟 내에 저장하는 것에 의해 데이터를 제조한다. 구성 데이터는 새로운 필드들의 개수, 새로운 필드들에 대한 데이터 유형들 및 값들, 그리고 새로운 필드들의 다른 특성들을 결정하기 위한 명령들을 제공한다.

[0193] 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 구성 데이터에 의해 제공된 정보를 이용하여 데이터를 제조한다. 정보는 생산 데이터 내의 현존 필드에 대한 데이터에 기초하여 제조될 새로운 필드를 위한 값들을 명시한다. 대안적으로, 이 정보는 임의의 현존 소스 데이터에 기초하지 않지만 대신 구성 데이터에 의해 명시되는 어떤 특성들에 따라 제조될 새로운 필드를 위한 값들을 명시한다.

[0194] 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 구성 데이터에 따라 하나 이상의 새로운 레코드들과 함께 생산 데이터 (26) 의 현존 소스 레코드들을 증강하기 위해 데이터 증강기 (20) 를 사용하는 것과 증강된 레코드들 (즉, 현존 소스 레코드들 및 새로운 레코드들 양쪽 모두) 을 타겟 내에 저장하는 것에 의해 데이터를 제조한다. 일부 실시예에서, 새로운 레코드들은 소스 레코드들과 동일한 레코드 포맷을 가진다.

[0195] 다른 실시예에서, 구성 데이터는 새로운 레코드들의 개수, 새로운 레코드들의 필드들에 대한 값들, 및 새로운 레코드들의 다른 특성들 중 하나 이상의 임의의 조합을 결정하기 위한 명령들을 제공한다. 이러한 예시들에 속하는 것은 구성 데이터가 스크래치로부터 제조될 새로운 레코드들 내의 하나 이상의 필드들을 위한 값들을 명시하는 것이다.

[0196] 다른 어떤 실시예에서, 구성 데이터는 프로파일을 명시하고 새로운 레코드들 내의 하나 이상의 필드들을 위한 값들이 그 프로파일을 충족시키도록 제조될 것을 요구한다. 이러한 일 실시예에서, 프로파일은 전체 레코드들 내의 특정한 필드를 위한 값들이 총괄하여 명시된 특성을 충족시킬 것을 명시한다. 특성의 예시는 특정한 평균 또는 특정한 분포를 가지는 값들이다. 예를 들어, 고객 데이터베이스 소스 내에서, 구성 데이터는 레코드들이 제조될 것을 요구하여 전체 레코드들에 걸친 "나이" 필드를 위한 값들이 특정한 평균을 가지는 프와송 분포 (Poisson distribution) 를 충족하도록 할 수도 있다.

[0197] 일부 실시예에서, 구성 데이터는 데이터 엔지니어링 모듈 (16) 이 데이터 생성을 위한 하나 이상의 방법을 적용할 것을 요구한다. 이러한 하나의 실시예에 대해, 데이터 엔지니어링 모듈 (16) 은, 하나 이상의 필드들에 대한 값들의 수정, 하나 이상의 새로운 필드들로 소스 레코드들을 증강, 및 하나 이상의 새로운 레코드들로 소스 레코드들을 증강의 방법들 중 임의의 조합을 적용한다.

[0198] 일부 실시예에서, 타겟은 제조된 레코드들만을 저장한다. 다른 실시예에서, 사용자는 소스를 명시하고 데이터 엔지니어링 모듈 (16) 은 특성에 기초하여 레코드들을 제조한다. 적절한 특성들의 예시는 소스의 레코드 포맷, 또는 소스의 하나 이상의 필드들의 프로파일이다.

[0199] 다른 실시예에서, 소스가 명시되지 않는다. 이러한 실시예에서, 데이터 엔지니어링 모듈 (16) 은 구성 데이터에 따라 스크래치로부터 자동적으로 레코드들을 제조한다.

[0200] 일부 실시예에서, 소스의 레코드 포맷은 타겟으로 매핑된다. 이러한 실시예에서, 구성 데이터는 소스의 레코드 포맷이 타겟에 의해 차용되어야 함을 지시한다. 다른 실시예에서, 구성 데이터는 소스의 레코드 포맷이 타겟으로 적용될 것과 새로운 레코드들이 소스의 레코드 포맷에 따라 데이터 엔지니어링 모듈 (16) 에 의해 스크래치로부터 제조될 것을 요구한다. 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 복수의 소스들에 의존하고 각

소스의 레코드 포맷은 부분적으로 또는 완전히 타겟으로 매핑된다. 적어도 하나의 실시예에서, 각 소스로부터의 관심 필드들의 포맷은 타겟으로 매핑된다.

[0201] 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 소스의 레코드 포맷을 타겟으로 매핑하고 그것을 수정한다. 이러한 예시에 속하는 것은 구성 데이터가 데이터 엔지니어링 모듈 (16) 이 필드의 이름을 변경하도록 하는 것과, 구성 데이터가 소스로부터의 필드의 삭제를 야기하는 것이다.

[0202] 데이터 엔지니어링 모듈 (16) 은 테스트 컴퓨터 (12) 상에서, 사용자가 데이터 소스를 식별하는 것을 가능하게 하기 위한 소스 윈도우를 가지는 사용자 인터페이스를 제공한다. 소스 윈도우는 사용자가 파일 또는 데이터베이스와 같은 소스 유형을 명시하는 것을 허용하는 소스 유형 메뉴와, 소스로의 또는 데이터베이스 소스를 위한 구성 파일로의 경로와 같은 소스의 식별자를 포함한다. 일부 실시예에서, 소스가 데이터베이스일 때, 사용자는 데이터베이스로부터 소스 데이터를 획득하기 위해 사용될 쿼리 (예를 들어, SQL 쿼리) 를 명시한다. 소스 윈도우는 사용자가 데이터 엔지니어링 모듈 (16) 이 새로운 레코드들을 제조할 것인가, 그렇다면 얼마나 많이 제조할 것인가를 지시하는 것을 허용하기 위한 옵션을 제공한다. 소스 윈도우는 사용자가 소스에 관한 다른 정보를 관측 또는 명시하는 것을 가능하게 한다. 예를 들어, 사용자는 소스의 레코드 포맷을 관측하고, 소스의 레코드 포맷을 정의하는 파일을 명시하고, 소스 데이터를 관측하고, 또는 소스 데이터의 프로파일을 관측할 수 있다.

[0203] 일부 실시예에서, 사용자 인터페이스의 소스 윈도우는 사용자가 데이터 엔지니어링 모듈 (16) 이 소스를 명시하지 않고 데이터를 제조하도록 하는 것을 허용한다. 특히, 소스 윈도우는 사용자가 소스 유형 메뉴 내의 소스 유형에 따른 제조된 데이터를 선택하는 것을 가능하게 한다. 소스 유형에 따른 제조된 데이터를 선택하는 것은 사용자 인터페이스 내의 데이터 생성 윈도우의 디스플레이를 야기한다. 데이터 생성 윈도우는 사용자가 데이터를 제조하기 위해 사용될 방법을 지시하고 제조될 새로운 레코드들의 개수를 지시하는 것을 가능하게 한다.

[0204] 사용자 인터페이스는 또한 사용자가 타겟을 식별하는 것을 가능하게 하는 타겟 윈도우를 제공한다. 타겟 윈도우 내의 타겟 유형 메뉴는 사용자가 타겟의 유형을 명시하는 것을 가능하게 한다. 타겟들의 예시는 파일 또는 데이터베이스를 포함한다. 타겟 윈도우는 또한 사용자가 타겟의 식별자 (예를 들어, 타겟 파일로의 경로 또는 타겟 데이터베이스를 위한 구성 파일로의 경로) 를 명시하는 것을 가능하게 한다. 타겟 윈도우는 소스 및 타겟이 식별되면 데이터 생성을 위한 다양한 구성 가능한 옵션들로의 엑세스를 사용자에게 제공하는 실행 버튼을 제공한다.

[0205] 데이터 엔지니어링 모듈 (16) 은 데이터를 제조하기 위한 여러 접근을 제공한다. 이들은 필드 수정, 필드 생성, 레코드 생성, 현존 소스의 이용, 및 부모 데이터세트의 사용을 포함한다. 이용 가능한 접근에 엑세스하기 위해, 사용자는 사용자 인터페이스의 데이터 생성 윈도우에 의존한다.

[0206] 필드 수정의 접근에서, 데이터 엔지니어링 모듈 (16) 은 소스 레코드들의 하나 이상의 필드들에 대한 값들을 수정한다. 일부 경우에, 데이터 엔지니어링 모듈 (16) 은 주어진 필드에 대한 전체 값들을 수정한다. 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 필드들의 값들을 수정하여 레코드들 전체에 걸친 주어진 필드의 값들의 분포가 타겟 분포와 매치하도록 한다. 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 주어진 필드에 대한 값들의 전체보다 더 적게 수정한다. 이러한 예시에 속하는 것은 데이터 엔지니어링 모듈 (16) 이 명시된 기준을 충족하지 않는 값들만을 수정하는 것이다. 예를 들어, 특정한 필드에 대해 허용된 값들의 특정한 범위를 벗어나는 임의의 값들이 수정될 수 있다.

[0207] 필드 생성의 접근에서, 데이터 엔지니어링 모듈 (16) 은 현존 레코드들에 대해 하나 이상의 새로운 필드들을 생성한다. 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 소스 데이터 내의 현존 필드에 대한 데이터에 기초하여 새로운 필드에 대한 값들을 제조한다. 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 스크래치로부터 새로운 필드에 대한 값들을 제조한다.

[0208] 레코드 생성의 접근에서, 데이터 엔지니어링 모듈 (16) 은 새로운 레코드들을 생성한다. 사용자는 새로운 레코드들의 개수 및 그들의 포맷 중 적어도 하나를 명시한다. 예를 들어, 타겟이 현존 소스 레코드들 및 새롭게 제조된 레코드들의 양쪽 모두와 함께 이주될 경우, 새로운 레코드들의 레코드 포맷은 소스 레코드들의 레코드 포맷과 동일하다. 만약 타겟이 단지 새롭게 제조된 레코드들과 함께 이주될 경우, 사용자는 제조된 레코드들에 적용될 레코드 포맷을 명시한다. 레코드 포맷은 필드들의 개수, 각 필드에 대한 데이터의 유형, 예를 들어 최댓값, 최솟값, 허용된 문자들의 세트 및 다른 특성들과 같은 각 필드에 대한 데이터의 특성들, 그리고 레코드 포맷의 다른 특징들을 포함한다.

[0209] 현존 데이터세트의 접근에서, 데이터 엔지니어링 모듈 (16) 은 현존 소스 레코드들 내의 각각의 키 값에 대해

명시된 개수의 새로운 레코드들을 제조한다. 키 값은 현존 소스 레코드 내의 관심 필드 내의 값이다.

[0210] 일 실시예에서, 보조 (auxiliary) 소스는 타겟 레코드들의 어떤 필드들을 이주시키기 위해 사용되는 데이터를 포함한다. 그러나 보조 소스는 소스 또는 타겟의 레코드 포맷과 매치하는 레코드 포맷을 가지지 않는다. 이 경우, 데이터 엔지니어링 모듈 (16) 은 보조 소스로부터의 하나 이상의 관심 필드들을 타겟 레코드들로 매핑한다. 부모 데이터세트의 접근에서, 소스는 계층 내의 부모 데이터세트이다. 이 경우, 데이터 엔지니어링 모듈 (16) 은 부모 데이터세트에 연관된 자식 데이터세트를 제조한다. 부모 데이터세트의 접근의 일 실시예에서, 소스로서 기능하는 부모 데이터세트는 고객 레코드들의 세트이고, 타겟으로서 기능하는 자식 데이터세트는 각각의 고객에 대한 하나 이상의 거래 레코드들의 세트이다. 키 필드는 자식 데이터세트 내의 레코드들을 부모 세트 내의 대응하는 레코드들로 연결한다. 예를 들어, "고객 ID" 필드는 고객 레코드들과 거래 레코드들을 연결하는 키 필드일 수 있다. 일부 경우에, 데이터 엔지니어링 모듈 (16) 은 얼마나 많은 자식 레코드들을 제조할지의 명세를 수신한다. 다른 경우에, 데이터 엔지니어링 모듈 (16) 은 자식 레코드들을 제조하기 위해 사용되지 않을 부모 레코드들의 백분율의 명세를 수신한다. 또 다른 경우에, 데이터 엔지니어링 모듈 (16) 은 자식 레코드들을 위한 레코드 포맷의 명세를 수신한다.

[0211] 일부 실시예에서, 데이터 엔지니어링 모듈 (16) 은 포맷 명세에 따라 데이터를 제조한다. 포맷 명세는 제조될 데이터의 포맷을 명시한다. 일 실시예에서, 포맷 명세는 제조될 데이터의 데이터 유형을 지시한다.

[0212] 다른 실시예에서, 데이터 엔지니어링 모듈 (16) 은 콘텐츠 기준에 따라 데이터를 제조한다. 콘텐츠 기준은 제조될 데이터의 특성들을 제한한다. 콘텐츠 기준의 예시는 값들의 허용된 범위, 최대 허용된 값, 및 허용된 문자들의 리스트를 포함한다.

[0213] 일부 경우에, 타겟 레코드들의 레코드 포맷은 포맷 명세 및 콘텐츠 기준을 명시한다. 다른 실시예에서, 사용자 인터페이스는 사용자가 필드에 대한 콘텐츠 기준 또는 포맷 명세와 같은 필드의 특성들을 명시하는 것을 가능하게 하는 필드 윈도우들을 제공한다.

[0214] 사용자 인터페이스는 사용자가 타겟 레코드 포맷을 수정하는 것을 가능하게 하는 레코드 포맷 윈도우를 더 포함한다. 이는 타겟의 하나 이상의 필드들에 대해 데이터 특성들을 수정하는 것을 포함할 수도 있다. 레코드 포맷 윈도우는 타겟 레코드 포맷 내의 필드들의 리스트를 디스플레이한다. 이러한 필드 리스트는 각 필드에 대한 데이터 유형을 또한 지시한다. 일부 실시예에서, 타겟 레코드 포맷 내의 필드들은 소스 레코드 포맷 내에서 또한 나타난다. 타겟 레코드 포맷 및 소스 레코드 포맷 양쪽 모두에서 나타나는 이러한 필드들은 필드 리스트 내에서 선택적으로 마크된다. 일부 실시예에서, 마크되지 않은 필드들은 타겟 레코드 포맷 내에서만 나타난다. 다른 실시예에서, 소스 레코드 포맷 내에서 나타나지만 타겟 레코드 포맷 내에서 나타나지 않는 필드들은 필드 리스트에 존재하지 않는다.

[0215] 레코드 포맷 윈도우는 사용자가 데이터 생성 특성들을 데이터 엔지니어링 모듈 (16) 로 통신하기 위해 타겟 레코드 포맷의 하나 이상의 필드들을 선택하는 것을 가능하게 한다. 사용자가 무엇이 선택되었는지의 추적을 유지하는 것을 보조하기 위해, 사용자 인터페이스는 타겟 레코드 포맷의 선택된 필드들의 선택 리스트를 포함한다. 선택 리스트에 나열된 필드들은 사용자가 데이터 생성 특성들을 명시하고자 하는 타겟 레코드 포맷의 필드들이다.

[0216] 일부 실시예에서, 선택 리스트는 타겟 레코드 포맷 내의 모든 필드들의 필드 리스트의 서브세트이다. 사용자가 타겟 레코드 포맷의 단지 일부의 필드들에 대해서만 데이터 생성 특성들을 명시하고자 할 경우에 이것이 발생한다.

[0217] 사용자 인터페이스는 사용자가 선택 리스트 내에 디스플레이된 선택된 필드들의 각각에 대해 레코드 포맷을 수정하는 것을 가능하게 한다. 예를 들어, 선택된 필드들의 각각에 대해, 사용자는 필드에 대한 데이터 유형의 지정, 필드에 대한 콘텐츠 생성 모드의 할당, 그리고 필드에 대한 데이터 특성들의 명시의 임의의 조합을 수행할 수 있다. 사용자 인터페이스는 선택된 필드들 각각에 대해 데이터 유형 윈도우, 콘텐츠 생성 윈도우, 및 데이터 특성 윈도우 중 하나 이상을 차례로 디스플레이한다. 이러한 윈도우들은 사용자가 선택된 필드들 각각에 대해 다양한 특징들을 명시하는 것을 가능하게 한다.

[0218] 상기 개시된 데이터 구동 테스트 프레임워크 (10) 는, 예를 들어, 적절한 소프트웨어 명령들을 실행하는 프로그래머블 컴퓨팅 시스템을 사용하는 적절한 소프트웨어를 실행하는 컴퓨팅 시스템을 사용하여 구현될 수 있거나 필드 프로그래머블 게이트 어레이 (FPGA) 와 같은 적절한 하드웨어 또는 어떤 하이브리드 형태로 구현될 수 있다. 예를 들어, 프로그래밍된 접근에서 소프트웨어는 하나 이상의 프로그래밍된 또는 프로그래머블 컴퓨팅 시스

템 (분산, 클라이언트/서버, 또는 그리드 등의 다양한 아키텍처일 수 있음) 에서 실행되는 하나 이상의 컴퓨터 프로그램의 절차들을 포함할 수 있으며, 각각은 적어도 하나의 프로세서, 적어도 하나의 데이터 저장 시스템 (휘발성 및/또는 비 휘발성 메모리 및/또는 저장 요소들을 포함함), (적어도 하나의 입력 디바이스 또는 포트를 사용하여 입력을 수신하고 적어도 하나의 출력 디바이스 또는 포트를 사용하여 출력을 사용하기 위한) 적어도 하나의 사용자 인터페이스를 포함한다. 소프트웨어는 예를 들어, 데이터 처리 그래프의 디자인, 구성 및 실행에 관련된 서비스를 제공하는 더 큰 프로그램의 하나 이상의 모듈을 포함할 수 있다. 프로그램의 모듈들 (예를 들어, 데이터 처리 그래프의 구성요소들) 은 데이터 저장소에 저장된 데이터 모델을 따르는 데이터 구조 또는 다른 조직화된 데이터로 구현될 수 있다.

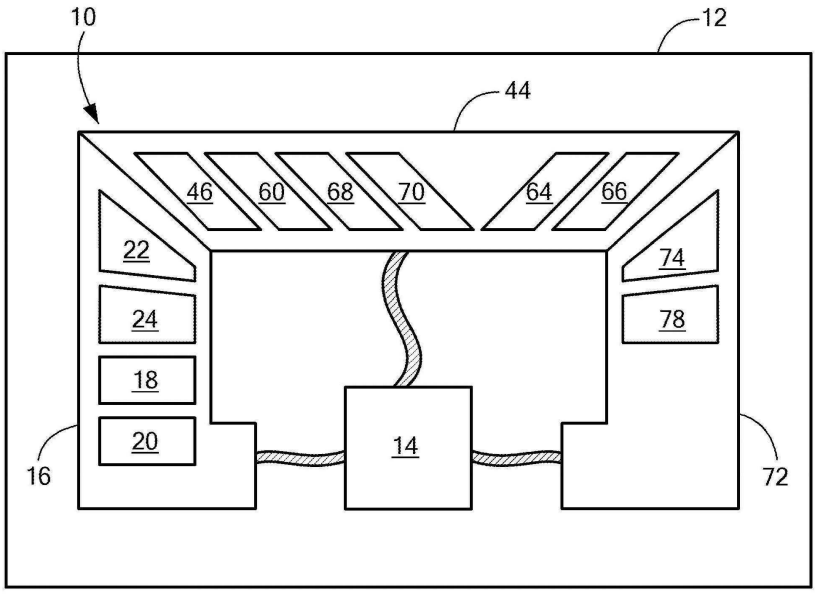
[0219] 소프트웨어는 시간의 기간 (예를 들어, 동적 RAM 과 같은 동적 메모리 디바이스의 리프레쉬 기간들 사이의 시간) 에 대한 매체의 물리적 속성 (예를 들어, 표면 피트 (pit) 들 및 랜드 (land) 들, 자구 (magnetic domain) 들, 또는 전하 (electrical charge)) 을 이용하여 휘발성 또는 비휘발성 저장 매체, 또는 임의의 다른 비 일시적 매체에 저장되는 것과 같이 비 일시적 형태로 저장될 수 있다. 명령들을 로딩하기 위한 준비로서, 소프트웨어는 CD-ROM 또는 다른 컴퓨터 판독가능 매체 (예를 들어, 범용 또는 특수 목적 컴퓨팅 시스템 또는 디바이스에 의해 판독 가능함) 같은 유형적, 비일시적 매체 상에서 제공될 수 있거나, 그것이 실행되는 컴퓨팅 시스템의 유형적, 비일시적 매체로 네트워크의 통신 매체를 통해 (예를 들어, 전파된 신호에 인코딩되어) 전달될 수 있다. 처리의 일부 또는 전부는 특수 목적 컴퓨터상에서 또는 코프로세서 또는 필드 프로그래머블 게이트 어레이 (FPGA) 들 또는 전용, 응용 주문형 집적 회로 (ASIC) 들과 같은 특수 목적 하드웨어를 사용하여 수행될 수 있다. 처리는 소프트웨어에 의해 명시된 계산의 상이한 부분들이 상이한 컴퓨팅 요소들에 의해 수행되는 분산 방식으로 구현될 수 있다. 각각의 이러한 컴퓨터 프로그램은 저장 디바이스 매체가 본원에서 설명된 처리를 수행하기 위해 컴퓨터에 의해 판독될 때 컴퓨터를 구성하고 운영하기 위해, 바람직하게는 범용 또는 특수 목적 프로그래머블 컴퓨터에 의해 액세스 가능한 저장 디바이스의 컴퓨터 판독가능 저장 매체 (예를 들어, 솔리드 스테이트 메모리 또는 매체, 또는 자기 또는 광 매체) 상에 저장되거나 다운로드 된다. 본 발명 시스템은 또한 컴퓨터 프로그램으로 구성된 유형적, 비일시적 매체로 구현되는 것으로 간주될 수 있고, 이렇게 구성된 매체는 본원에서 설명된 처리 단계들 중 하나 이상을 수행하기 위해 컴퓨터로 하여금 특정한 미리 정의된 방식으로 동작하게 한다.

[0220] 본 발명의 다수의 실시예들이 설명되었다. 그럼에도 불구하고, 전술한 설명은 예시를 위한 것이고 다음 청구항들의 범위 내에서 정의되는 본 발명의 범위를 제한하는 것이 아니다. 따라서, 다른 실시예들 역시 아래의 청구항들의 범위 내에 있다. 예를 들어, 다양한 변형들이 본 발명의 범위를 벗어나지 않고 이루어질 수 있다. 더 나아가, 상기 개시한 단계들 중 일부는 순서 독립적일 수 있고 따라서 설명된 것과 다른 순서로 수행될 수 있다.

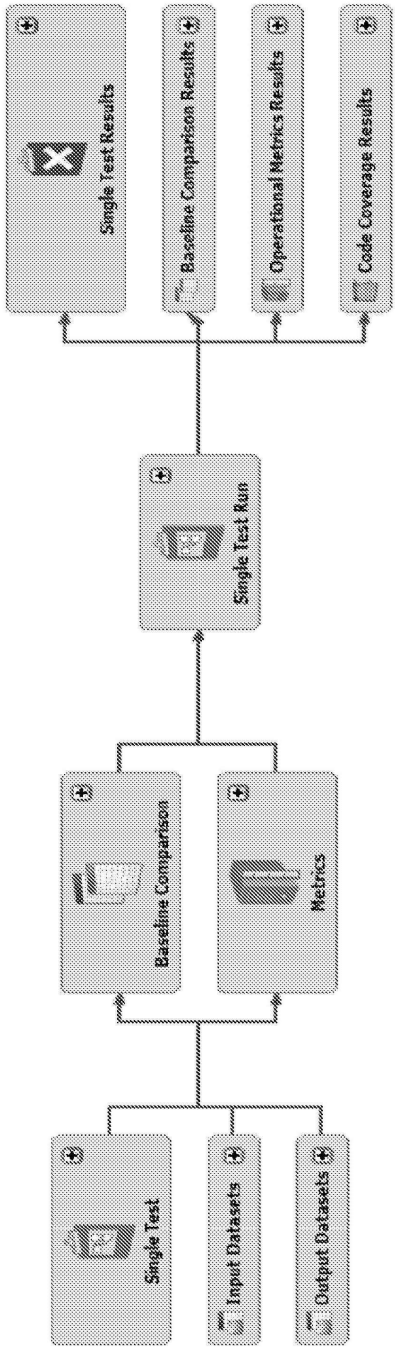
[0221] 청구항들은 본 발명을 설명하고 그 바람직한 실시예이며 특허로서 보호되고 새로운 것으로서 주장된다.

도면

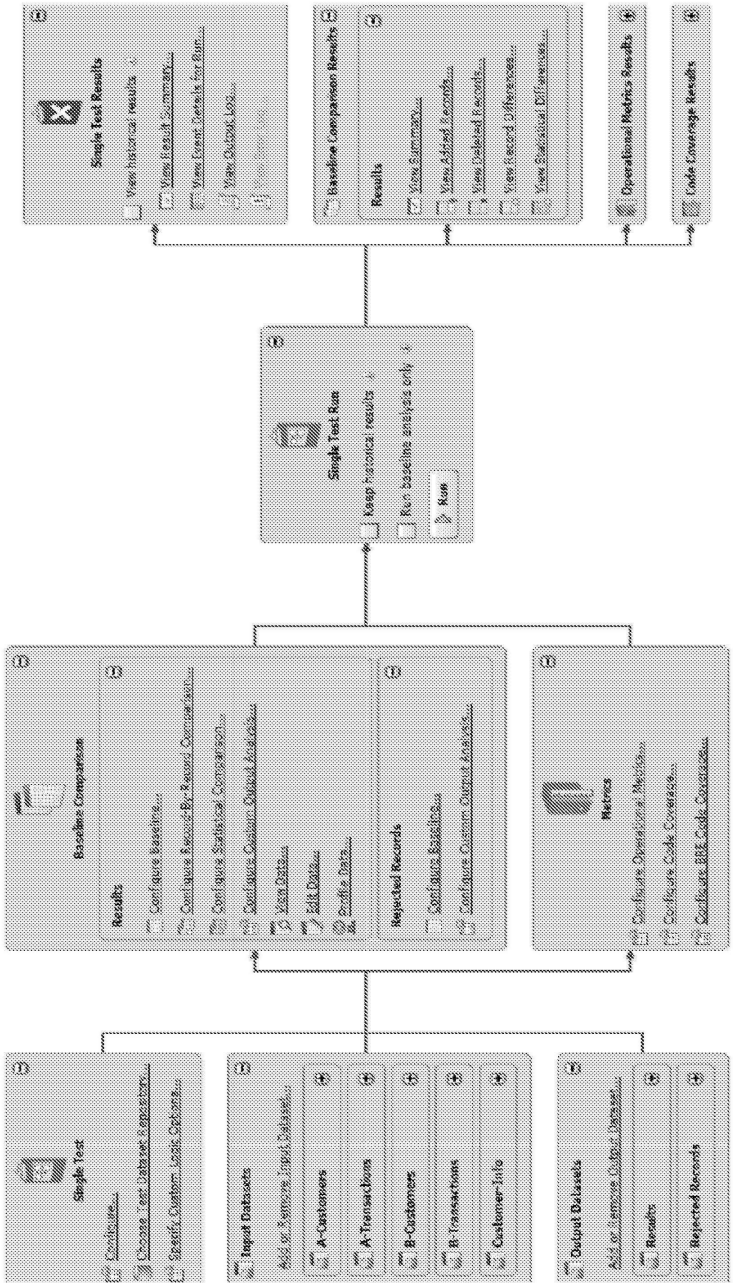
도면1



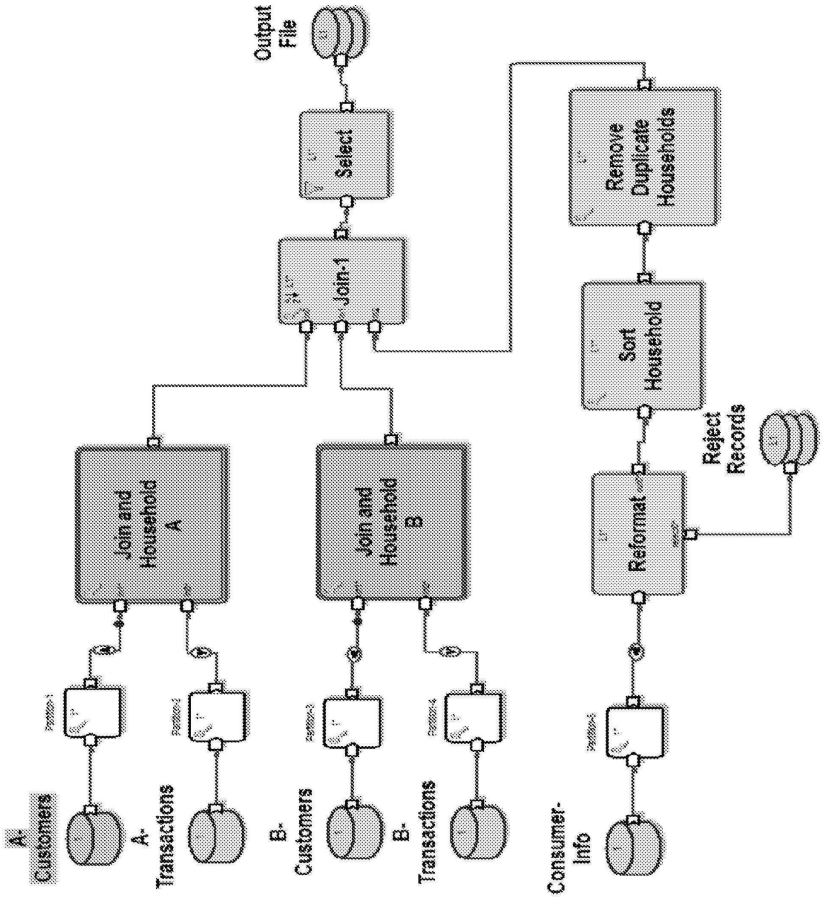
도면2


Single Test Results
Baseline Comparison Results
Operational Metrics Results
Code Coverage Results
Single Test Run
Baseline Comparison
Metrics
Single Test
Input Datasets
Output Datasets

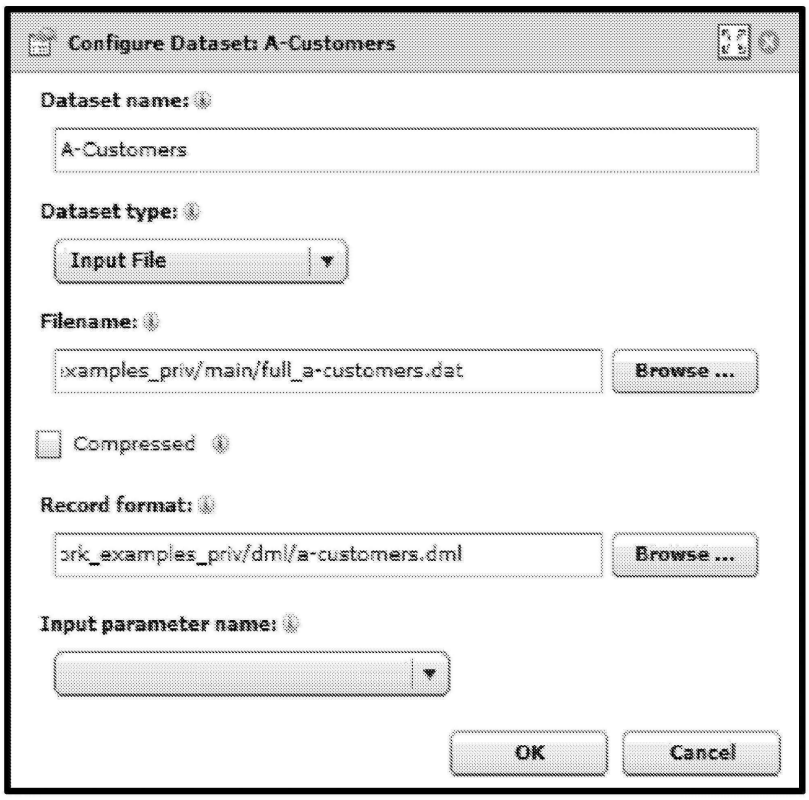
도면3


Simple Test Results
View historical results
View result summary...
View event details for Run...
View output log...
Baseline Comparison Results
Results
View summary...
View added records...
View deleted records...
View record differences...
View statistical differences...
Operational Metrics Results
Code Coverage Results
Simple Test Run
Keep historical results
Run baseline analysis only
Run
Baseline Comparison
Results
Configure baseline...
Configure Record-By-Record Comparison...
Configure Statistical Comparison...
Configure Custom Output Analysis...
View Data...
Edit Data...
Profile Data...
Rejected Records
Configure baseline...
Configure Custom Output Analysis...
Metrics
Configure Operational Metrics...
Configure Code Coverage...
Configure BB Code Coverage...
Simple Test
Configure...
Choose Test Dataset Template...
Specify Custom Tests Options...
Input Datasets
Add or Remove Input Datasets...
A-Customers
A-Transactions
B-Customers
B-Transactions
Customer-Info
Output Datasets
Add or Remove Output Datasets...
Results
Rejected Records

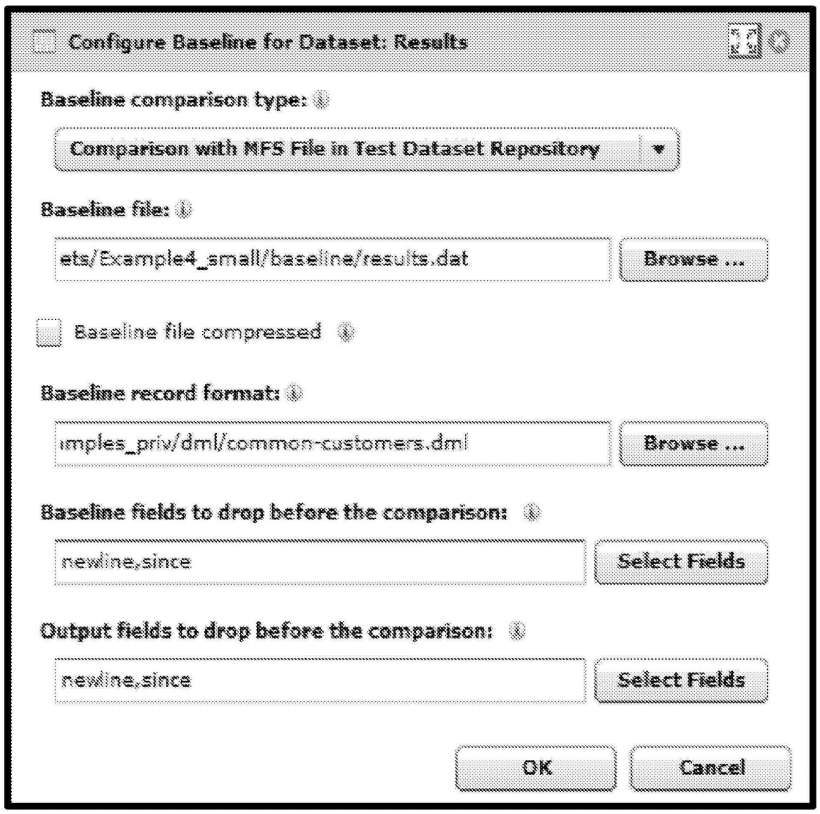
도면4


Output File
Select
Join-1
Remove Duplicate Households
Sort Household
Reject Records
Reformat
Join and Household A
Join and Household B
A-Customers
A-Transactions
B-Customers
B-Transactions
Consumer-Info

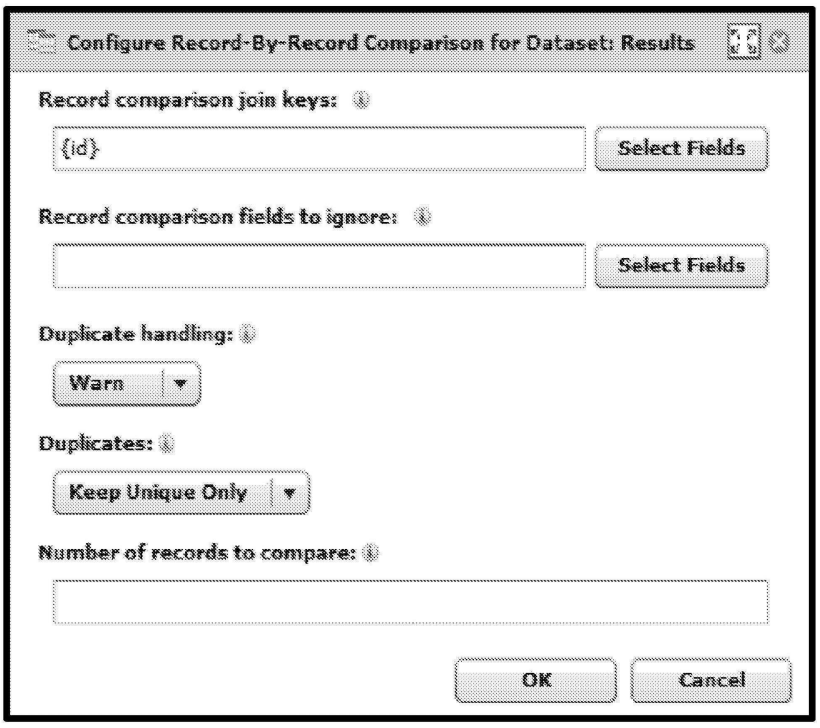
도면5


Configure Dataset: A-Customers
Dataset name:
A-Customers
Dataset type:
Input File
Filename:
xamples_priv/main/full_a-customers.dat
Browse ...
Compressed
Record format:
ork_examples_priv/dml/a-customers.dml
Browse ...
Input parameter name:
OK
Cancel

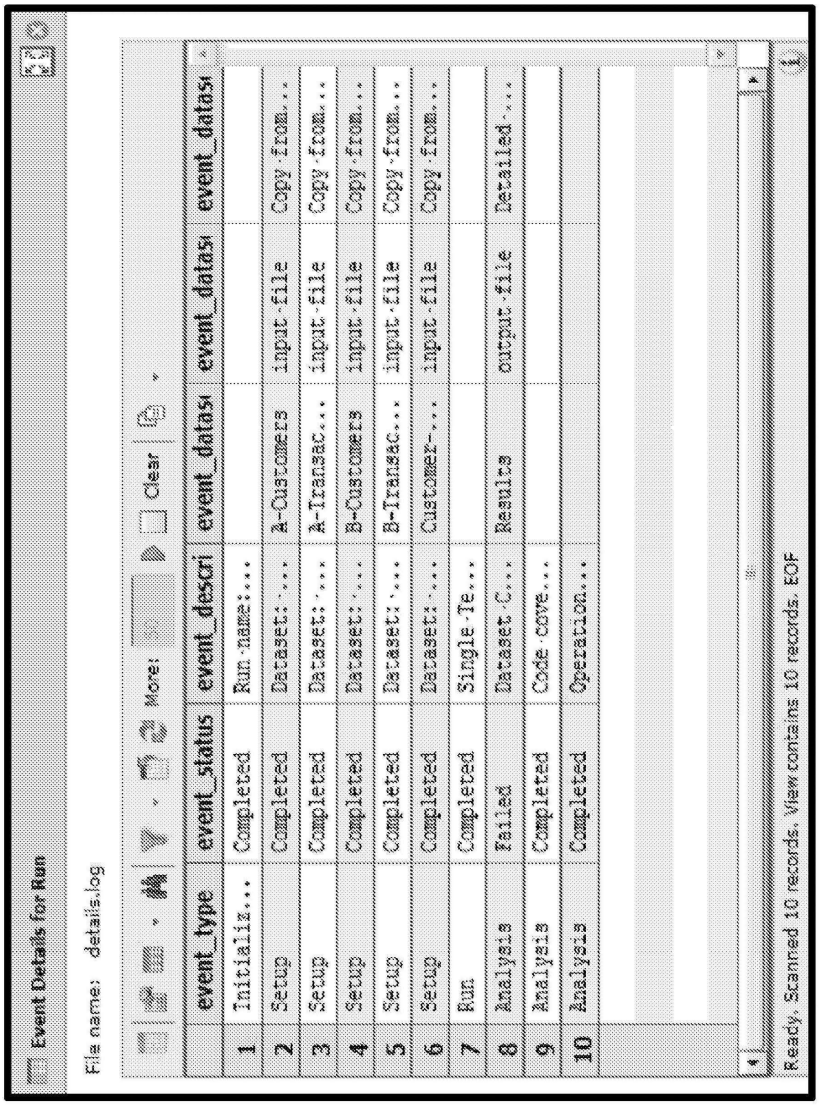
도면6


Configure Baseline for Dataset: Results
Baseline comparison type:
Comparison with MFS File in Test Dataset Repository
Baseline file:
ets/Example4_small/baseline/results.dat
Browse ...
Baseline file compressed
Baseline record format:
mples_priv/dml/common-customers.dml
Browse ...
Baseline fields to drop before the comparison:
newline,since
Select Fields
Output fields to drop before the comparison:
newline,since
Select Fields
OK
Cancel

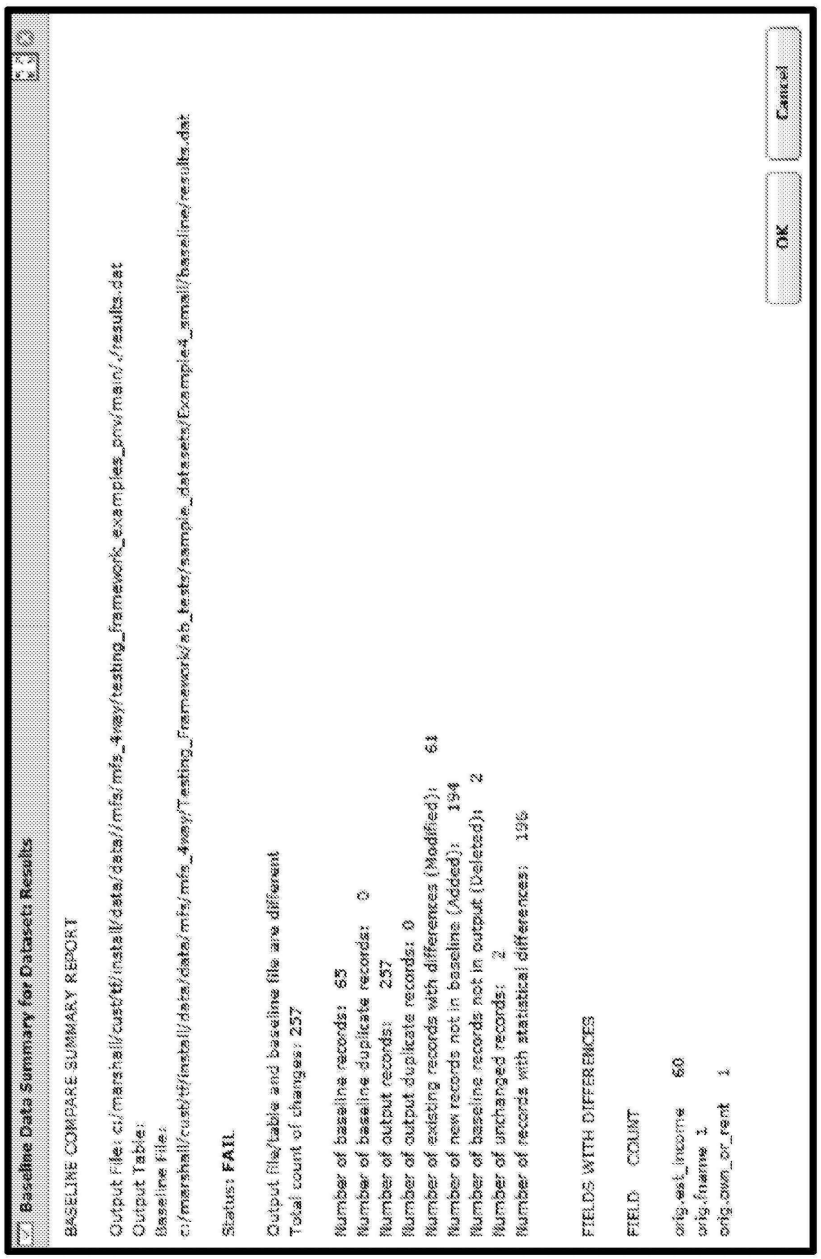
도면7


Configure Record-By-Record Comparison for Dataset: Results
Record comparison join keys:
{id}
Select Fields
Record comparison fields to ignore:
Select Fields
Duplicate handling:
Warn
Duplicates:
Keep Unique Only
Number of records to compare:
OK
Cancel

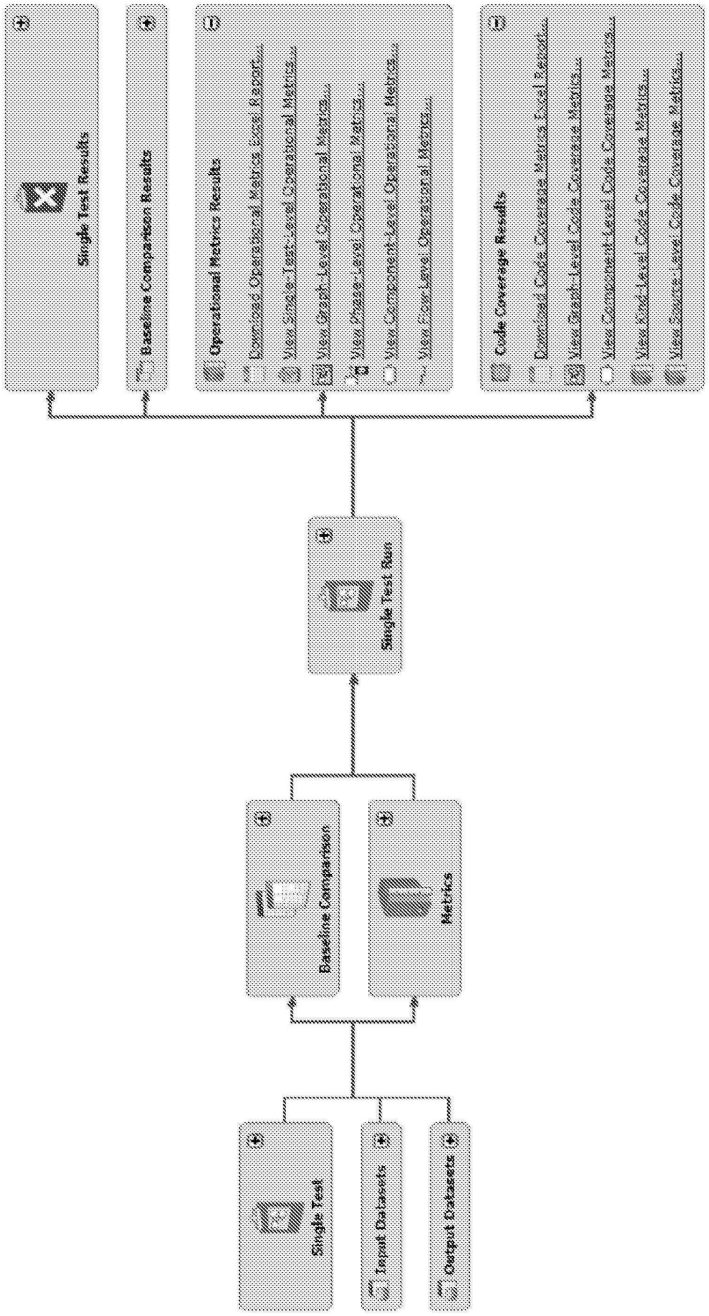
도면8


Event Details for Run
File name: details.log
More:
Clear
event_type
event_status
event_descri
event_datas
event_datas
event_datas
1 Initializ... Completed Run name:...
2 Setup Completed Dataset:... A-Customers input file Copy from...
3 Setup Completed Dataset:... A-Transac... input file Copy from...
4 Setup Completed Dataset:... B-Customers input file Copy from...
5 Setup Completed Dataset:... B-Transac... input file Copy from...
6 Setup Completed Dataset:... Customer-... input file Copy from...
7 Run Completed Single Te...
8 Analysis Failed Dataset C... Results output file Detailed ...
9 Analysis Completed Code cove...
10 Analysis Completed Operation...
Ready. Scanned 10 records. View contains 10 records. EOF

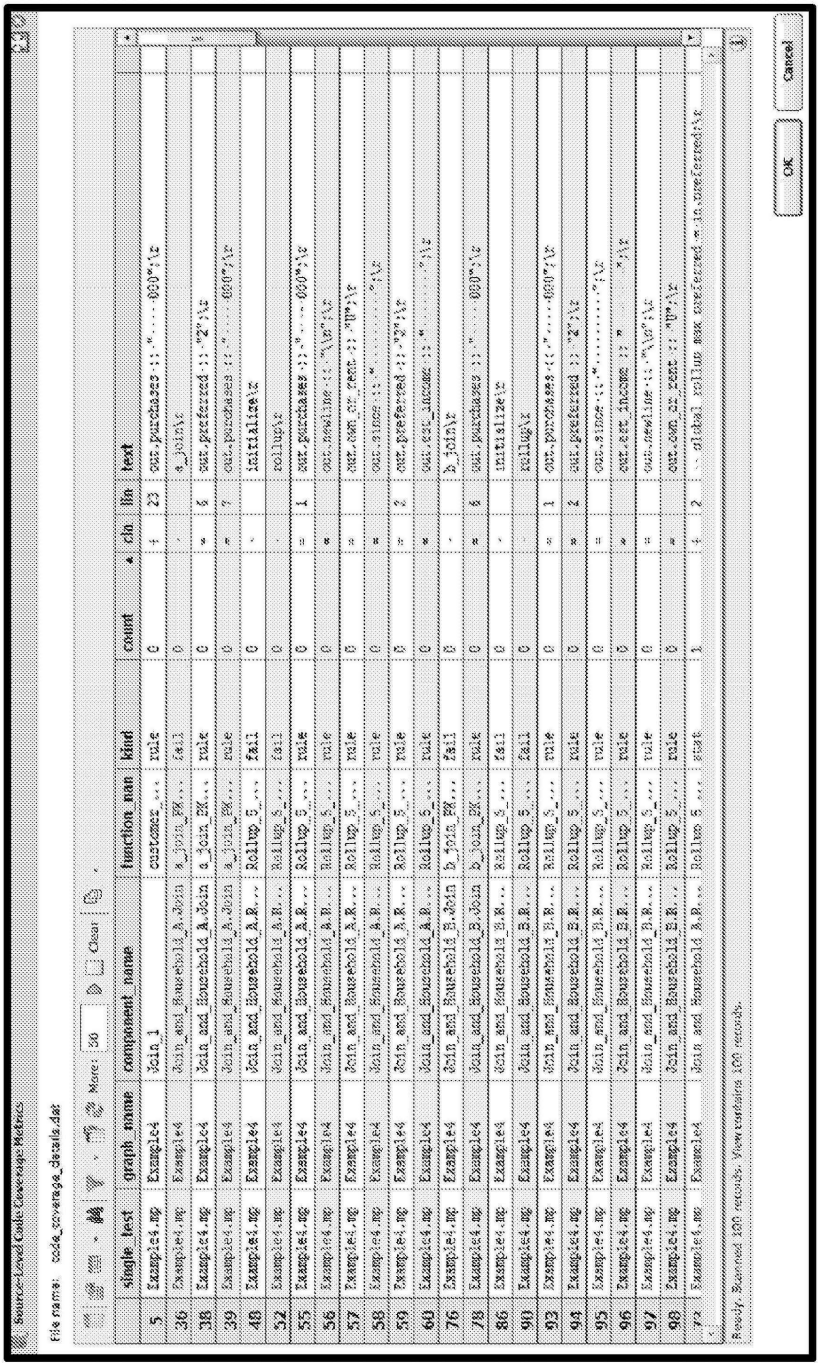
도면9


Baseline Data Summary for Dataset: Results
BASELINE COMPARE SUMMARY REPORT
Output File: c:/marshall/custtt/install/data/data/mfs/mfs_4way/testing_framework_examples_pnw/main/./results.dat
Output Table:
Baseline File:
c:/marshall/custtt/install/data/data/mfs/mfs_4way/Testing_Framework/ab_tests/sample_datasets/Example4_small/baseline/results.dat
Status: FAIL
Output file/table and baseline file are different
Total count of changes: 257
Number of baseline records: 65
Number of baseline duplicate records: 0
Number of output records: 257
Number of output duplicate records: 0
Number of existing records with differences (Modified): 61
Number of new records not in baseline (Added): 194
Number of baseline records not in output (Deleted): 2
Number of unchanged records: 2
Number of records with statistical differences: 196
FIELDS WITH DIFFERENCES
FIELD COUNT
orig_est_income 60
orig_fname 1
orig_own_or_rent 1
OK
Cancel

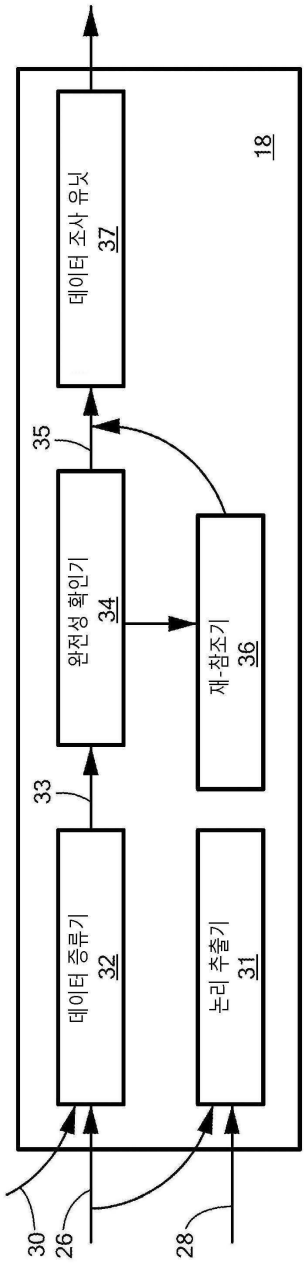
도면10


Single Test Results
Baseline Comparison Results
Operational Metrics Results
Download Operational Metrics Excel Report...
View Single-Test Level Operational Metrics...
View Graph-Level Operational Metrics...
View Phase-Level Operational Metrics...
View Component-Level Operational Metrics...
View Flow-Level Operational Metrics...
Code Coverage Results
Download Code Coverage Metrics Excel Report...
View Graph-Level Code Coverage Metrics...
View Component-Level Code Coverage Metrics...
View Unit-Level Code Coverage Metrics...
View Source-Level Code Coverage Metrics...
Single Test Run
Baseline Comparison
Metrics
Single Test
Input Datasets
Output Datasets

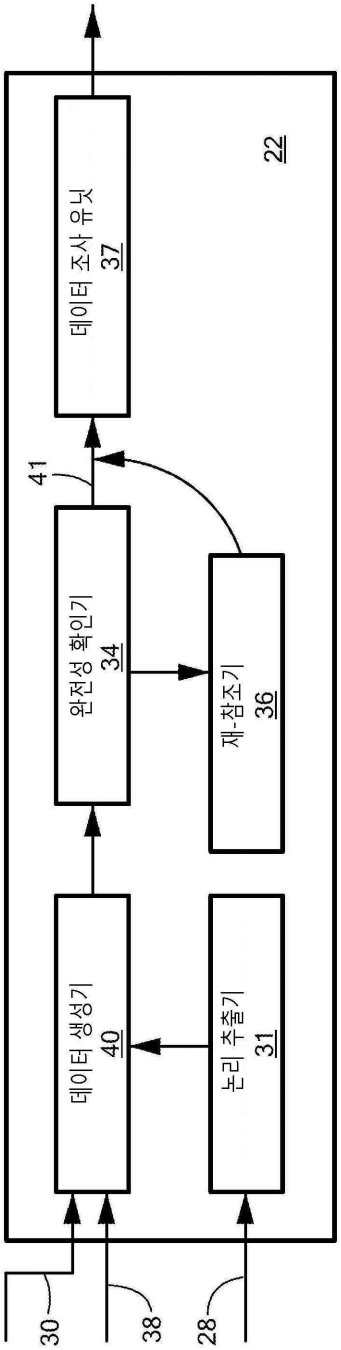
도면11


Source-Level Code Coverage Metrics
File name: code_coverage_details.dat
More: 50
Clear
single_test
graph_name
component_name
function_nam
kind
count
ch
lin
text
OK
Cancel
Ready. Scanned 100 records. View contains 100 records.

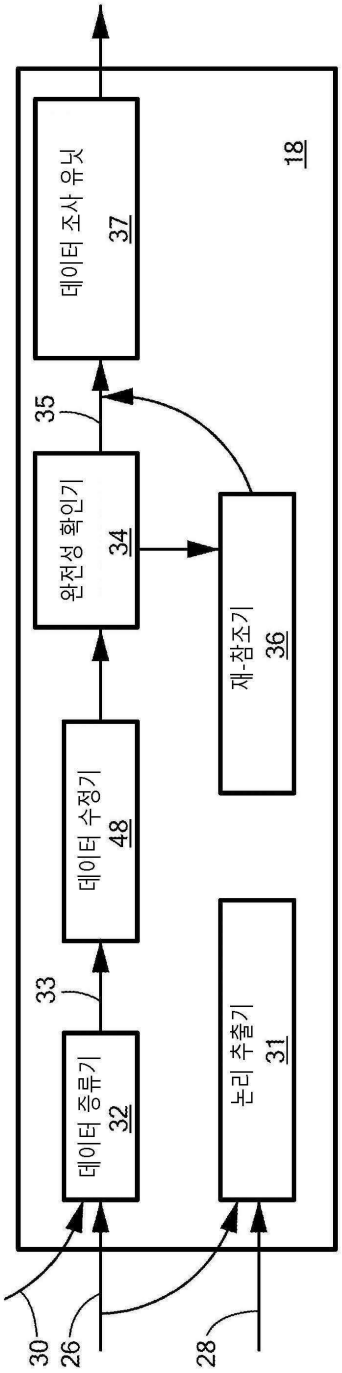
도면12


18
데이터 조사 유닛
37
35
완전성 확인기
34
재-참조기
36
33
데이터 증류기
32
논리 추출기
31
30
26
28

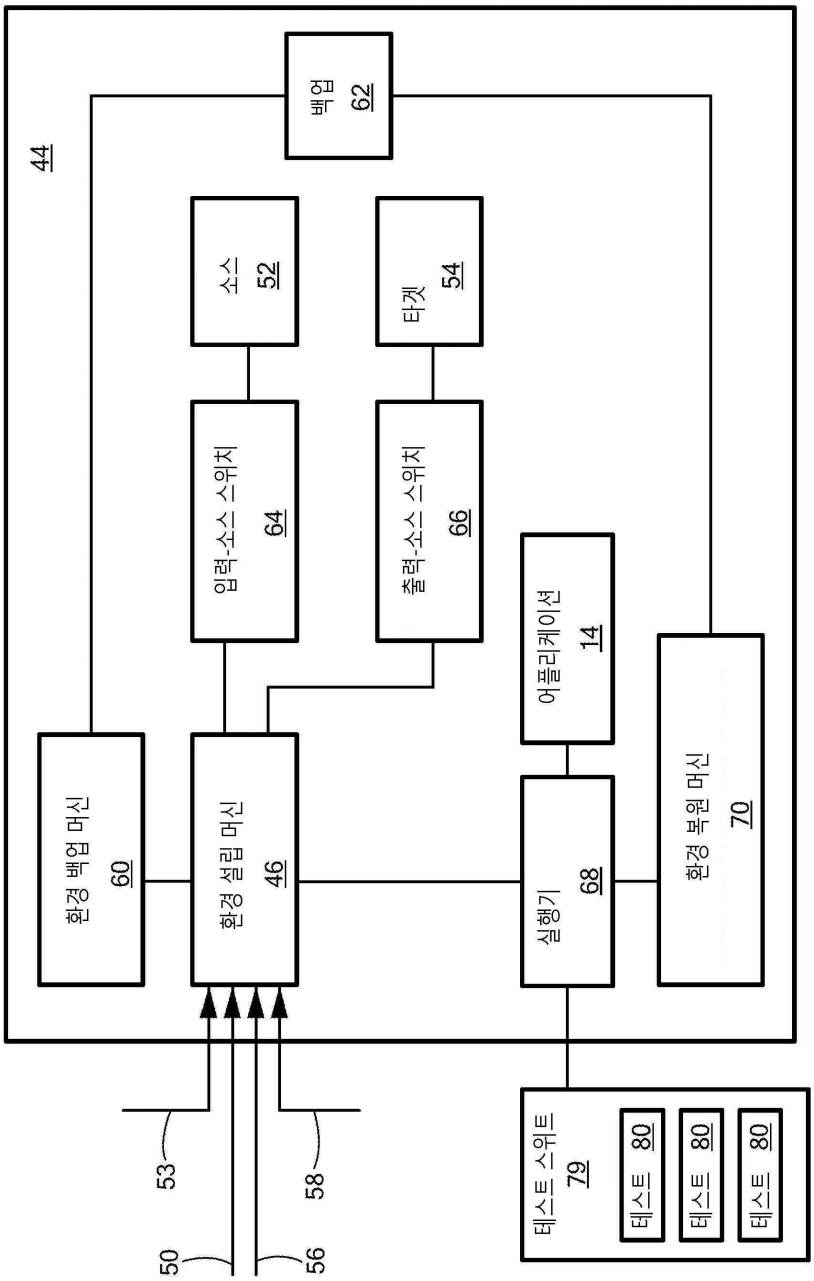
도면13



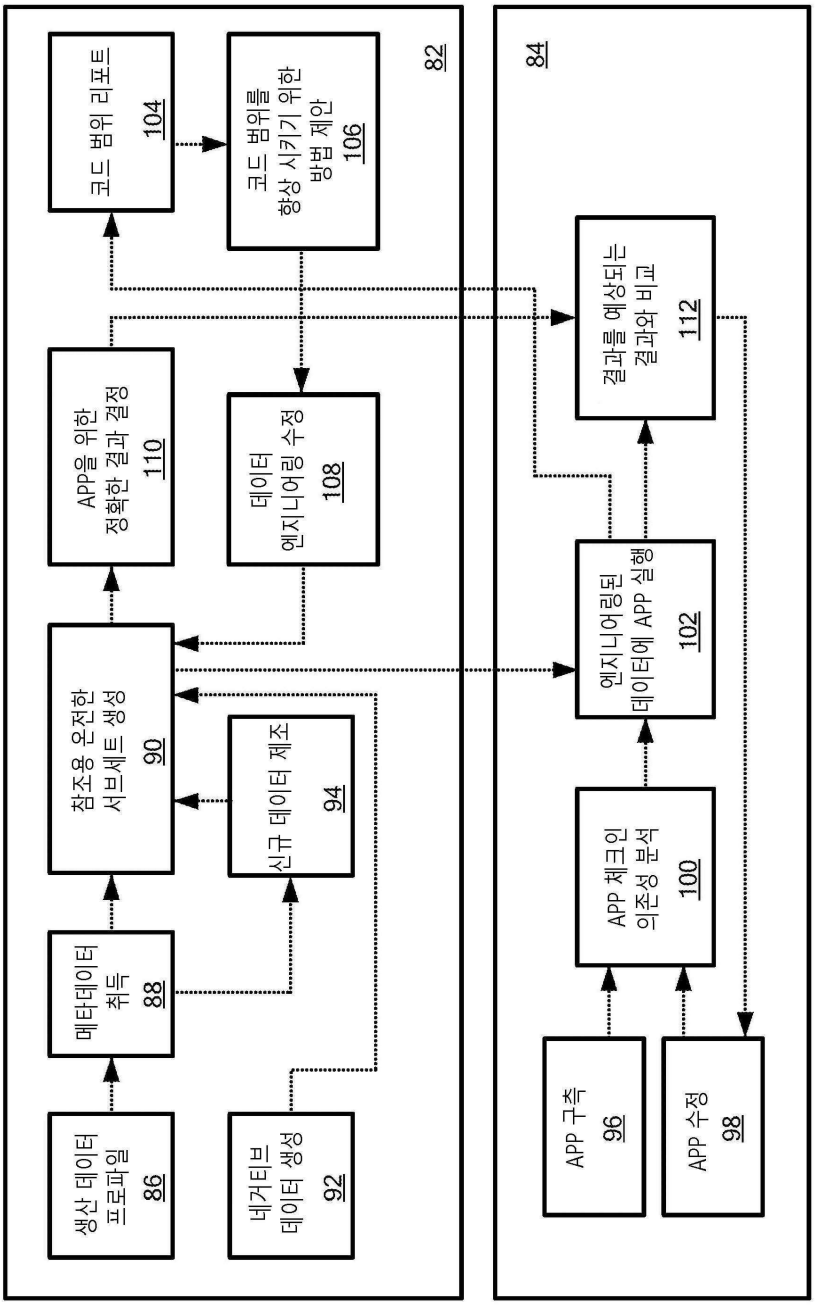
도면14


18
데이터 조사 유닛
37
35
완전성 확인기
34
재-참조기
36
데이터 수정기
48
33
데이터 증류기
32
논리 추출기
31
30
26
28

도면15


44
백업 62
소스 52
타겟 54
입력-소스 스위치 64
출력-소스 스위치 66
어플리케이션 14
환경 백업 머신 60
환경 설립 머신 46
실행기 68
환경 복원 머신 70
테스트 스위트 79
테스트 80
테스트 80
테스트 80
53
58
50
56

도면16


82
84
생산 데이터 프로파일 86
메타데이터 취득 88
참조용 온전한 서브세트 생성 90
네거티브 데이터 생성 92
신규 데이터 제조 94
APP 구축 96
APP 수정 98
APP 체크인 의존성 분석 100
엔지니어링된 데이터에 APP 실행 102
코드 범위 리포트 104
코드 범위를 향상 시키기 위한 방법 제안 106
데이터 엔지니어링 수정 108
APP을 위한 정확한 결과 결정 110
결과를 예상되는 결과와 비교 112