(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년12월27일
(11) 등록번호 10-1099173
(24) 등록일자 2011년12월20일

(51) Int. Cl.
G06F 9/44 (2006.01) *G06F 9/45* (2006.01)
(21) 출원번호 10-2006-7008844
(22) 출원일자(국제출원일자) 2004년10월14일
심사청구일자 2009년10월14일
(85) 번역문제출일자 2006년05월08일
(65) 공개번호 10-2006-0120670
(43) 공개일자 2006년11월27일
(86) 국제출원번호 PCT/US2004/034013
(87) 국제공개번호 WO 2005/048100
국제공개일자 2005년05월26일
(30) 우선권주장
10/964,899 2004년10월13일 미국(US)
60/518,285 2003년11월07일 미국(US)
(56) 선행기술조사문헌
US06381742 B2*
US06407753 B1*
US20030088857 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
소니 일렉트로닉스 인코포레이티드
미국, 뉴저지 07656, 파크 리지, 원 소니 드라이브
(72) 발명자
설름, 제프리, 타이
미국 94087 캘리포니아주 써니베일 더블유. 레밍톤 드라이브 803
레하, 빅터, 글렌
미국 94538 캘리포니아주 프레몬트 랄스톤 케몬 4052
(뒷면에 계속)
(74) 대리인
주성민, 이중희, 백만기

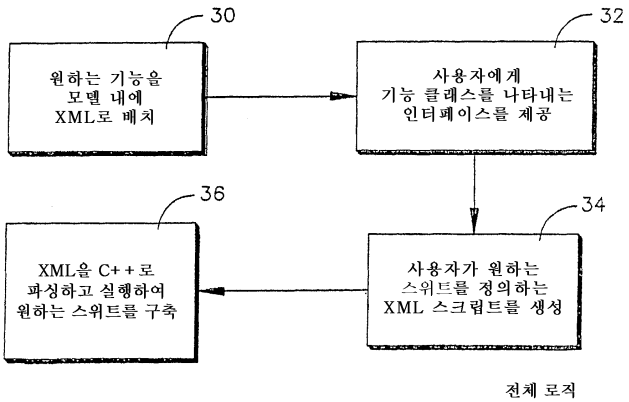
전체 청구항 수 : 총 9 항 심사관 : 유병철

(54) 소프트웨어 스위트를 구축하기 위한 시스템 및 방법

(57) 요 약

본 발명은, 컴퓨터를 위한 완전한 소프트웨어 스위트를 어셈블링하는 정식 소프트웨어 프로그래밍 기술에 대한 필요성이 없이, 쉽게 사용되고 이해될 수 있는 XML 기반 프로그래밍 언어, 툴키트, 및 개발 환경(24, 26, 28)을 제공한다.

대 표 도 - 도3



전체 로직

(72) 발명자

팻톤, 스코트, 애버리

미국 92025 캘리포니아주 에스콘디도 썬셋 드라이브 1955

캘리안푸, 비자야난드, 무랄리드하

미국 94089 캘리포니아주 써니베일 와일드우드 배뉴 넘버 98 1235

특허청구의 범위

청구항 1

적어도 2개의 컴퓨터 애플리케이션을 컴퓨터(22)용 소프트웨어 패키지로 어셈블링하기 위한 방법으로서,

XML 구성체(construct)들을 사용자에게 제공하는 단계 - 상기 구성체들은 애플리케이션을 얻기 위하여 사용될 수 있는 객체 지향 프로그래밍 환경 내의 클래스들임 - 와,

상기 사용자가 상기 구성체들을 이용하여 XML 스크립트를 구성하도록 하는 단계 - 상기 스크립트는, 상기 스크립트를 구성하기 위하여 상기 사용자에 의해 사용된 상기 구성체에 의해 얻을 수 있는 애플리케이션들을 포함하기 위하여, 상기 소프트웨어 패키지의 내용(contents)을 정의함 - 와,

C++ 소프트웨어 코드로 되게(rendering) 하기 위해 상기 스크립트를 파싱(parsing)하는 단계와,

상기 스크립트에 의해 정의된 애플리케이션들을 상기 소프트웨어 패키지로 자동 어셈블링하기 위하여 상기 C++ 소프트웨어 코드를 실행하는 단계

를 포함하는, 어셈블링 방법.

청구항 2

삭제

청구항 3

제1항에 있어서,

사용자가 선택하도록 컴퓨터 디스플레이(40) 상의 클래스 윈도우(42) 내에 적어도 일부 클래스를 제공하는 단계를 포함하는, 어셈블링 방법.

청구항 4

제1항에 있어서,

상기 C++ 소프트웨어 코드를 실행하는 단계를 개시하기 위해 적어도 상기 소프트웨어 패키지의 식별과 관련된 정보의 입력을 상기 사용자에게 요청하는 단계를 포함하는, 어셈블링 방법.

청구항 5

적어도 2개의 컴퓨터 애플리케이션을 컴퓨터용 소프트웨어 패키지로 어셈블링하기 위한 시스템으로서,

소프트웨어 애플리케이션들의 리스트를 생성하는데 유용한 객체 지향 애플리케이션 프로그래밍 인터페이스들(API)을 포함하는 모델 컴포넌트(24);

상기 애플리케이션들을 패키지로 자동 어셈블링하기 위해, 상기 모델 컴포넌트(24)와 통신하고, 상기 리스트를 실행을 위한 코드로 파싱하는 파서(parser)를 포함하는 제어기 컴포넌트(26); 및

상기 리스트 생성 시 사용을 위해, 애플리케이션을 얻기 위하여 사용될 수 있는 객체 클래스들을 사용자에게 제공하기 위하여, 상기 제어기 컴포넌트(26)와 통신하는 뷰 컴포넌트(28)

를 포함하는, 어셈블링 시스템.

청구항 6

제5항에 있어서,

상기 리스트는 상기 애플리케이션들과 관련된 저장 위치들을 포함하는, 어셈블링 시스템.

청구항 7

제5항에 있어서,

상기 API는 XML 기반인, 어셈블링 시스템.

청구항 8

제7항에 있어서,

상기 코드는 C++인, 어셈블링 시스템.

청구항 9

제8항에 있어서,

사용자가 선택하도록 컴퓨터 디스플레이(40)상의 클래스 윈도우(42) 내에 적어도 일부 클래스를 제공하는 수단을 포함하는, 어셈블링 시스템.

청구항 10

제9항에 있어서,

적어도 상기 패키지의 식별과 관련된 정보의 입력을 상기 사용자에게 요청하는 수단을 포함하는, 어셈블링 시스템.

명 세 서

[0001] 관련 출원

[0002] 본 출원은 2003년 11월 7일에 제출된 미국가특허출원 일련번호 제60/518,285호의 우선권을 주장한다.

기 술 분 야

[0003] 본 발명은 일반적으로 퍼스널 컴퓨터에 관한 것이다.

배 경 기 술

[0004] 소니의 VAIO® 컴퓨터와 같은 퍼스널 컴퓨터는 각 프로젝트 빌드(project build)를 위한 사양(specification)으로 작성된 맞춤형(custom) 소프트웨어 컴포넌트의 세트를 포함한다. 즉, 일부 컴퓨터는 제1 소프트웨어 스위트, 가령, 오디오-비디오 소프트웨어와 함께 워드 프로세서를 구비해야 하는 반면, 다른 그룹의 컴퓨터들은 제2의 상이한 소프트웨어 스위트를 구비하도록 특정되어 구매자에게 보다 많은 선택을 제공할 수 있다. 여기서, "소프트웨어 스위트(software suite)"는 컴퓨터의 모든 소프트웨어 세트뿐만 아니라, 가령, 공장 및 테스팅 팀에 대한 컴포넌트 릴리즈(release)를 의미하는데, 여기서 컴포넌트는 전체 프로젝트 릴리즈의 일부로서 구성된다.

[0005] 각각의 맞춤형 스위트를 작성하는 데에는, 다수의(multiple) 및 비연속(disjoint) 프로그램과 관련된 많은 단계를 필요로 한다. 지금까지는 스위트의 다양한 프로그램을 어셈블링할 때, 엔지니어가 원하는 프로그램을 중앙 데이터베이스 또는 데이터베이스들로부터 가령, 해당 소프트웨어를 컴퓨터에 로딩하기 위한 디스크상에 수동으로 배치 및 복사해야 했다. 이는 시간이 소요되며, 스크래치로부터 새롭게 특정된 스위트를 구축하기 위한 수동적 개입을 필요로 한다. 더욱이, 빌드가 자동화되지 않기 때문에, 그러한 "빌드들(builds)"에 불가피하게 에러 및 불일치가 생긴다.

[0006] 따라서, 여기서 자세히 알 수 있다시피, 컴퓨터의 그룹을 위해 소프트웨어 스위트를 어셈블링하는 자동화된 방법을 제공하는 것이 바람직하다. 그러나, 여기서 더 알 수 있다시피, 어셈블러는 C++과 같은 프로그래밍 언어에 대해 전문적 기술을 가지고 있지 않을 수 있다. 따라서, 본 발명은 정식의 프로그래밍 지식 없이도 소프트웨어 스위트를 어셈블링하는 자동화된 방법을 제공할 필요성을 인식하고 있다.

발명의 상세한 설명

[0007] 컴퓨터용 소프트웨어 패키지를 어셈블링하는 방법은, 사용자에게 XML 구성체(constructs)를 제공하는 단계 및 사용자가 이 구성체를 사용하여 XML 스크립트를 구성하도록 하는 단계를 포함하고, 스크립트는 소프트웨어 패키

지의 내용을 정의한다. 또한, 본 방법은 스크립트를 파싱(parsing)하여 C++ 소프트웨어 코드가 되도록 하는 단계와, C++ 소프트웨어 코드를 실행하여 그 내용을 소프트웨어 패키지로 자동 어셈블링하도록 하는 단계를 포함한다.

[0008] 바람직한 실시예에서, 구성체는 객체 지향 프로그래밍 환경 내의 클래스이다. 클래스는 사용자가 이를 선택하도록 컴퓨터 디스플레이 상의 클래스 윈도우에서 사용자에게 제공될 수 있다. 실행 동안, 사용자는 가령, 소프트웨어 패키지의 식별과 관련된 정보에 대해 촉구받을 수 있다(prompt).

[0009] 다른 양태로서, 적어도 두 개의 소프트웨어 애플리케이션을 컴퓨터상으로의 로딩을 위한 패키지로 자동 어셈블링하는 시스템은, 스크립트 내에서 애플리케이션을 식별하기 위한 계층적 객체 지향 수단을 포함한다. 또한, 시스템은 스크립트를 실행가능한 코드로 파싱하는 수단을 포함한다. 패키지를 자동 어셈블링하기 위해 실행가능한 코드를 실행하기 위한 수단이 제공된다.

[0010] 또 다른 양태로서, 소프트웨어 시스템은, 소프트웨어 애플리케이션의 리스트를 생성하는데 유용한 객체 지향 애플리케이션 프로그래밍 인터페이스(Application Programming Interface: API)를 포함하는 모델 컴포넌트를 포함한다. 제어기 컴포넌트는 모델 컴포넌트와 통신하며, 소프트웨어 애플리케이션 리스트를 그 실행을 위한 코드로 파싱하는 파서(parser)를 포함하여 애플리케이션을 패키지로 자동 어셈블링한다. 소프트웨어 애플리케이션 리스트 생성 시에, 객체 클래스들을 사용하도록, 뷰 컴포넌트(view component)가 제어기 컴포넌트와 통신하여 사용자에게 객체 클래스들을 제공한다.

실 시 예

[0016] 본 발명의 상세, 즉 그 구조 및 동작 모두는, 유사한 참조 번호가 유사한 부분을 나타내는 첨부 도면을 참조하여 가장 잘 이해될 수 있다.

[0017] 먼저, 도 1을 참조하면, 전반적으로 참조 번호 10으로 표시되는, 마우스, 키보드 등과 같은 하나 이상의 입력 디바이스(14)를 구비한 로드 정의 컴퓨터(load definition computer)(12)와 컴퓨터 모니터, 프린터, 네트워크 등과 같은 하나 이상의 출력 디바이스(16)를 포함할 수 있는 시스템이 도시되어 있다. 로드 컴퓨터(12)는 로드 데이터베이스(18)와 같은 하나 이상의 소프트웨어 애플리케이션의 데이터 소스와 통신하여, 이들 애플리케이션을 가령 Sony VAIO® 컴퓨터일 수 있는 랩탑 컴퓨터(22)와 같은 타겟 컴퓨터상에 소프트웨어 애플리케이션을 로딩하기 위한 가령, 광학 디스크(20)에 복사될 수 있는 스위트 또는 패키지로 어셈블링한다.

[0018] 도 2는 로드 컴퓨터(12)에 의해 실행될 수 있는 소프트웨어의 아키텍쳐를 나타내고, 도 3은 아키텍쳐의 주요 기능을 비한정적인 플로우 차트 포맷으로 나타낸다. 도 2에 도시된 바와 같이, 본 소프트웨어는 모델 컴포넌트(24), 제어기 컴포넌트(26), 및 뷰 컴포넌트(28)를 포함할 수 있다. 모델 컴포넌트(24)는 시스템 애플리케이션 프로그래밍 인터페이스(Application Programming Interface: API), 바람직하게는 소프트웨어 애플리케이션의 리스트를 생성하는데 유용한 XML 객체 지향 구성체를 포함한다. 따라서, API는 프로그램 자동화를 위한 통상의 Windows 커맨드를 제공하는 함수의 집합이다. 모델 컴포넌트(24)는 본질적으로 파일 조작 및 프로그램 실행으로부터 메세지 디스플레이 및 데이터베이스(18)에 대한 액세스에 이르는 함수를 포함하는 툴킷(toolkit) 및 정보 저장소(information repository)이다.

[0019] 보다 구체적으로, 모델 컴포넌트(24)는, 이하 개시되는 내용과 같이, Microsoft사의 ".NET" 시스템에 의해 이해될 수 있는 동적 링크 라이브러리(Dynamic Link Library: DLL) 파일을 생성한다. 또한, 모델 컴포넌트(24)는 애플리케이션을 얻기 위해 사용자에 의해 선택될 수 있는 모든 XML 객체 클래스를 위한 저장소이다. 또한, 모델 컴포넌트(24)는, "int" 및 "char*"와 같은 프리미티브 C++ 데이터 유형 구성체가 Object* 및 String*으로 각각 변환되도록, 일반적인 랩퍼(wrapper) 함수를 위한 어댑터(adapter)를 구비하는 제어부를 포함할 수 있다. 또한, 모델 컴포넌트(24)는, 임의의 커맨드 실행의 결과를 로그(log)하는데 사용될 수 있는 설정(setting) 클래스를 가질 수 있다. 이는 실제로 클래스 내의 C++ 함수에 의해 구현될 수 있다. 각 API 커맨드는 단일 클래스 내의 함수일 수 있거나, 개별 클래스로서 구현될 수 있다.

[0020] 제어기 모듈(26)은 변수 선언(variable declaration) 및 조건문(conditional statements)을 포함하는 시스템 언어 구성체를 비롯하여 모든 비지니스 로직(business logic)을 포함하고, 사용자로부터 수신된 애플리케이션의 리스트를 그 실행을 위한 코드로 파싱하여 애플리케이션을 패키지로 자동 어셈블링하는 파서를 대표한다. 이런 목적을 위해, 제어기 모듈(26)은 뷰 컴포넌트(28) 및 모델 컴포넌트(24) 양자간에 최소한의 커플링(coupling)을 포함하여, 그 둘 사이를 명료하게 구분한다. 제어기 모듈(26)은, 다른 두 컴포넌트 중 어느 하나로 코드가 변

경되는 경우, 재컴파일링(recompilation)을 필요로 하지 않는다.

[0021] 뷰 모듈(28)은, 패키지로 어셈블링된 애플리케이션을 정의하는 리스트 또는 스크립트를 생성할 때, 그 사용을 위해, 제어기 컴포넌트(26)를 통한 모델 컴포넌트(24)에 대한 액세스를 허용하여 객체 클래스를 사용자에 제공하는 사용자 인터페이스이다. 두 개의 뷰가 존재할 수 있는데, 하나는 단순히 커맨드 라인으로부터 실행될 수 있는 프로그램 실행 및 디버깅(debugging)을 위해 사용되고, 다른 하나는 시스템 스크립트의 생성, 편집, 및 실행을 위해 사용되는 사용자 인터페이스이다. 양자는 입력을 XML 스크립트로부터 또는, 부가적으로 인터페이스 내의 API 선택을 통해 수신할 수 있다.

[0022] 도 3은 시스템(10) 내에서 구현되는 전체 로직을 나타낸다. 블럭(30)에서 개시하여, 가령 XML 기반 객체 클래스 내에서의 원하는 기능을 모델 컴포넌트(24) 에 둔다. 블럭(32)에서, 뷰 컴포넌트(28)는 가령, 도 1에 도시된 모니터(16)상에, 이하 더 논의될 기능 클래스를 사용자에게 제공하기 위해 호출된다. 블럭(34)에서, 사용자는 소프트웨어 패키지 또는 스위트로 어셈블링된 애플리케이션의 스크립트 또는 리스트를 생성하기 위해 다양한 클래스를 선택할 수 있다. 일단 완료되면, 로직은 블럭(36)으로 이동하여, C++과 같은 실행가능한 코드로 XML을 파싱한 후에 그 코드를 실행하여, 스크립트 내에 포함된 명령어(가령, 임의의 애플리케이션이 발견될 수 있는 위치)에 따라, 스크립트에서 식별된 애플리케이션을 자동 검색하고 패키지로 어셈블링함으로써 스크립트를 실행할 수 있다. 실행의 일부로서, 사용자는 변수명, 변수값, 및 가령, 소프트웨어 패키지명 등의 다른 정보에 대해 촉구받을 수 있다.

[0023] 도 4는 상기 원리에 따라 생성된 스크립트(38)가 계층적일 수 있고, 결과적으로 실행가능한 프로그램을 형성하기 위해 결합되는 계층적인 일련의 커맨드로서 취급될 수 있다는 것을 나타낸다. 스크립트 내의 모든 커맨드는, 유리하게도, 문서 유형 정의(Document Type Definition), 즉 DTD로 지칭되는 모든 가능한 커맨드의 마스터 파일로 검증될 수 있다. 실행 전에 DTD에 대해 XML 스크립트 내의 모든 커맨드를 검증함으로써, 구문(syntax)이 정확하도록 보장된다.

[0024] 전술한 바와 같이, 가령, C++로의 XML 파싱은, 모든 시스템 언어 구성체를 핸들링하는(handle) 제어기 컴포넌트(26) 내에서 수행된다. 일부 실시예에서, 스크립트 검증(validation)은, 검증을 위해 시작부터 끝까지 XML 구문을 한번에 한 노드씩 메모리 내로 판독입력하는 Microsoft .net 시스템 API 클래스 XMLValidatingReader를 사용하여 핸들링될 수 있다. 실제 파싱은, W3C 문서 객체 모델(Document Object Model), 즉 DOM[3]을 이용하는 .NET API 클래스 XPathNavigator를 이용하여 수행될 수 있다. XML 코드에 대해 순방향 파싱만 허용하는 XMLTextReader와는 달리, DOM은 역방향 탐색(navigation)도 허용한다. 대부분의 기본적인 시스템 커맨드의 경우, 순방향 파싱만으로도 충분하지만, 조건문 또는 루핑 중 어느 하나를 필요로 하는 진보형 커맨드(advanced command)에서는, DOM 형의 파싱을 구현하고 전체 코드를 메모리 내에 유지하는 역방향 파싱도 요구된다.

[0025] 따라서, 커맨드의 두 유형, 즉 기본형 및 진보형이 제공될 수 있다. 기본형 커맨드는 모델 컴포넌트(24)로부터 본래 그대로 사용될 수 있다. 이들은 임의의 언어 구성체에 독립적일 수 있고, 사실상 대부분의 시스템 API를 구축할 수 있다. 한편, 진보형 커맨드는, 동일한 커맨드를 여러 번 호출하는 것이 필요할 수 있는 부가적인 XML 파싱을 필요로 한다. 또한, 조건문 및 루핑문과 같은 일부 진보형 커맨드는 네스트 커맨드(nested commands)를 허용한다. 도 4에 도시된 XML 스크립트(38)의 계층적 구조는 전반적으로 나무(tree)와 유사하다. 스크립트 레이아웃에 따라, 구조는 낮은 깊이이면서 하나 이상의 관목(shrubs)을 나타낼 수 있고, 또는 높은 깊이이면서 나무 또는 숲(forest)을 나타낼 수 있다. 사실, 도 4는 조건문뿐만 아니라 XML 스크립트 자신에 적용되는 네스트 기능(nested capability)을 나타낸다. 도 4의 다이아그램은 스크립트에서 나타날 수 있는 것과 동일한 방식으로 표현되며, 좌측에서 시작하여 우측으로, 위에서 시작하여 아래로 판독된다.

[0026] 일부 실시예에서, 네스트 커맨드를 통해 순방향 및 역방향 탐색을 추적하기 위해, 깊이 및 현재 노드가 공지되어야 한다. 스크립트 실행 동안, 원하는 경로를 결정하기 위해 리턴 값이 기록될 수 있다. 원하지 않는 경로는 폐기된다. 이는 스택의 세트 내에 정보를 기록함으로써 행하여 진다. 노드에 대한 현재의 포인터는, 조건문을 파싱함으로써 트리 속으로 더 깊이 이동한다. 조건문을 실행한 후 현재 깊이는 스택상에서 푸쉬(push)된다. 완료 후, 포인터가 네스트 조건문을 통해 더 깊이 또는 더 얕게 이동하기 때문에, 현재 깊이는 스택으로부터 각각 푸쉬(push)되거나 팝(pop)된다. 또한, 조건마다 리턴값을 추적하기 위한 유사한 스택이 존재한다. 대체로, 조건문마다 4 개의 스택이 존재할 수 있는데, 깊이를 추적하는 것과, If문 또는 Else문 양자에 대해 리턴값을 추적하는 것이 존재할 수 있다. 현재 노드의 값을 스택 내의 현재 값과 비교함으로써, 시스템(10)은 가장 복잡한 네스트 구조도 이해할 수가 있다.

[0027] 본 발명은 클래스 구조가 공지되어야 하는 두 가지 상황이 발생할 수 있다는 것을 알고 있다. 첫 번째는, 모든

시스템 API 커맨드의 명칭과 파라미터를 디스플레이하도록 사용자 인터페이스에 의해 요구된다. 두 번째는, 실행 동안 입력을 전달하고 각 커맨드를 발생하도록, 시스템 API 커맨드 파라미터를 동적으로 해석하는 제어기 모듈(26)에 의해 요구된다.

[0028] 도 5는 시스템 API의 리스트(42)(본질적으로, 기능 클래스)가 우측 구획(right pane)에 제공되고, 예시적인 메세지 박스 커맨드에 대한 파라미터가 하부 구획(44)에 제공되어 있는 예시적 사용자 인터페이스(40)를 나타낸다. 또한, 다른 커맨드에 대한 파라미터는 우측 구획을 상향 또는 하향으로 스크롤링함으로써 하부 구획 내에 표시될 수 있다. 또한, 유리하게도 도구 모음(46)이 제공될 수 있다. 또한, 메인 구획(50)도 제공될 수 있다.

[0029] 각 시스템 API 커맨드는 제어기(26) 내에서 커맨드마다 가변수의 파라미터를 핸들링하기 위한 개별적 파싱 함수의 요구를 필요치 않고, 오히려 .NET API를 사용하는 보편적인 파싱 함수를 공유하여 시스템 API 커맨드를 동적으로 해석하고 호출할 수 있다. 이는 인트로스펙션(introspection) 및 동적 호출(invocation)로 지칭되는 객체 지향 컴포넌트 개념을 통해 이루어질 수 있다. XML 스크립트로부터 취해진 입력은, 동적 호출을 위해 시스템 API 커맨드에 동적으로 전달된다. 이는 입력, XML 스크립트가 제어기 컴포넌트(26)를 재컴파일할 필요 없이 변할 수 있다는 것을 의미한다. 통상, 이들 개념을 적용하지 않고도, 파라미터 값은 통상의 애플리케이션을 통해 정적 호출에 대해 고정될 수 있다. 진보형 커맨드만이 명시적, 개별적 파싱 함수를 필요로 한다.

[0030] .NET 내부 프로시저 호출(internal procedure calls)은 앞서 언급되었다. 직접적 또는 간접적 참조(reference) 중 어느 하나를 통한 시스템 API는 전체적으로 모델 컴포넌트(24) 내에 포함된다. 모델 컴포넌트(24) 내에 직접적으로 포함되지 않은 모든 API 커맨드는, 제어기 컴포넌트(26)를 그 각각의 위치로 향하게 하는 랩퍼 함수를 가지지 않으면 안된다. 몇몇 경우에 있어서, 거의 또는 전혀 변형이 없이, 이미 존재하는 코드 및 프로그래밍 노력을 재사용하는 많은 네스트 랩퍼 함수가 필요할 수 있다. 몇몇 실시예에서, 언어 독립형 코드 재사용은 Microsoft사의 컴포넌트 객체 모델(Component Object Model: COM)에 의해 용이하게 될 수 있다.

[0031] 비한정적 실시예에서, 로직을 추가하는 것을 필요로 하는 어떠한 사용자 인터페이스도 존재하지 않는다. 일단 컴포넌트가 등록되면, 마치 코드가 모델 컴포넌트(24) 내의 관련 클래스 내에 직접적으로 존재하는 것처럼, 사용하도록 이용가능하다. 도 1에 도시된 데이터베이스(18) 내로의 데이터 입력은 그 태스크에 특정되는 GUI 필드를 포함하는 사용자 인터페이스를 구비한 COM 객체에 의해 용이하게 될 수 있다. 이는 단일 함수 단편(single function piece)일 수 있지만, 단독으로 완전한 애플리케이션은 아니며, 오히려 객체, 가령 사용자 인터랙션 및 백엔드 기능을 포함하는 애플리케이션의 단편이다. 이는 본래 그대로 사용될 수 없기 때문에, 사용 이전에 컨테이너(container) 내에 배치되어야 한다. 본 시스템(10)은 ControlForm이라 지칭되는 컨테이너를 제공한다. 이 클래스는 기본적으로 두 개의 버튼, 즉 OK 및 Cancel을 구비한 윈도우이다. 실제 기능은, ControlForm 컨테이너 내에 배치되는, 하나 이상의 상호교환가능한 COM 객체로부터 유래한다. 그 다형성(polymorphism)의 예로서, 컨테이너 객체는, 일례에서는 데이터베이스(18) 로그인 데이터 입력을 요구하는 윈도우이고, 다른 예에서는 데이터베이스(18) 프로젝트 선택 데이터 입력을 요구하는 윈도우이다. 상호교환가능한 컴포넌트를 디스플레이하기 위한 일 컨테이너의 사용은 보편적인 컨테이너 로직과, OK 및 Cancel 버튼이 각각의 COM 컴포넌트 내에 있을 필요가 없다는 것을 의미한다. 또한, 필요한 경우, 둘 이상의 컴포넌트가, 고유한, 새로운 폼 및 컴포넌트를 특별히 생성해야 하지 않고도, 동일한 폼 상에서 디스플레이될 수 있다는 것을 의미한다.

[0032] 본 발명의 원리에 따르면, 각 시스템 API 커맨드는, 커맨드가 올바르게 실행되었는지 여부를 알려주는 값을 리턴하여, 부울 참(true) 또는 거짓(false)에 기초하여 진보된 구조를 파싱하는 것이 가능하게 하는 것이 바람직하다. 요구된 부울 리턴 값에 더하여, 각 커맨드는 거의 무제한 수의 커맨드 특정 값을 리턴할 수 있다. .NET ArrayList 구조는 데이터를 동적 확장가능한 Object의 어레이로서 저장함으로써 이를 가능하게 한다. Object는 어떤 다른 유형으로의 변환을 허용하는 일반적인 .NET 구성체일 수 있다.

[0033] ArrayList 리턴 구조는 메모리 내에 일시적으로만 유지될 수 있다. XML 스크립트로부터 실행되는 각각의 커맨드의 경우, 리턴 구조는 다음 커맨드의 리턴 구조로 대체된다. 이는 시스템 API 커맨드를 실행한 후 즉시 수행되는 임의의 리턴 값들의 저장을 필요로 한다. 진보형 시스템 커맨드를 실행할 때, 이 프로세스는 자동으로 행해진다. 사용자 정의된 변수를 사용하는 경우, 사용자는 변수에 대한 메모리로부터의 리턴 값을 수동으로 저장할 수 있다.

[0034] 일부 실시예에서, 사용자 정의된 변수를 선언하는 4가지 방법이 제공될 수 있다. 첫째는 시스템 API 커맨드 AddVariable을 통하는 것인데, 이는 변수 명 및 변수값 모두가 런 타임 전에 스크립트 내에 배치되는 것을 요구

한다. 각 변수 유형은 캐릭터의 스트링(string)으로서 저장되고, 모든 커맨드 내의 모든 파라미터는, 나중에 현재 시스템 커맨드에 의해 또는 다른 시스템 커맨드를 통해 다른 유형으로 변환될 수 있는 스트링으로서, 초기에 판독될 수 있다.

[0035] 변수를 선언하는 두 번째 방법은, 시스템 API 커맨드 PromptAddVariable을 통하는 것인데, 이는 AddVariable과 유사하지만, 실행 동안 사용자에게 변수값에 대해 촉구한다. 변수명은 여전히 스크립트 내에서 선언되고 런 타임에서 고정된다.

[0036] 변수를 선언하는 세 번째 방법은, 시스템 API 커맨드 AddVarFromMem을 통하는 것인데, 이는 이전 커맨드의 ArrayList 리턴 구조 내의 특정 위치에 기초하여 리턴값을 저장한다. 이는 이전 커맨드 및 이용가능한 리턴 구조에 대한 약간의 지식을 필요로 한다.

[0037] 변수 AddMultipleFromMem를 선언하기 위한 네 번째 방법은, AddVarFromMem과 유사하지만, 이전 커맨드로부터의 모든 리턴값을 다수의 사용자 정의된 변수 내로의 저장을 허용한다.

[0038] AddVarFromMem 또는 AddMultipleFromMem 중 어느 하나를 사용함으로써, 프로그램이 구동되는 동안 일시적으로 저장된 리턴값이 메모리 내에 유지될 수 있다. 전술한 4개의 커맨드의 조합을 사용함으로써, 사용자는 변수를 선언 및 할당할 수 있고, 변수 내로 사용자 입력을 판독입력할 수 있으며, 변수를 다른 커맨드의 출력에 할당할 수 있다.

[0039] 시스템 언어 특정 커맨드는 "If", "For", 및 "While"을 포함할 수 있다. 시스템 API 커맨드는 CopyFolder, DeleteFolder, RenameFolder, CopyFile, DeleteFile, RenameFile, ExecuteProgram, AddRegKey, RemoveRegKey, CreateFile, WriteToFile, AddIniSection, RemoveIniSection, AddIniKey, RemoveIniKey, Settings, SetStatus, MsgBox, IsFile, IsDir, IsInFile, IsRegKey, IsRegValue, IsIniSection, IsIniKey, IsNT를 포함할 수 있다.

[0040] 이하, 정식 XML 포맷(formatting) 없이, 스크립트를 나타내는 소위 "유즈 케이스(Use Case)"가 제공된다.

[0041] 1.1 INI 구성 파일 생성

Description Create INI file or files for a given recovery tool
Use Case identifier B1
Author
Date 5/01/2003
Revised
Actors Release Engineer
Pre-conditions FI-%Project name%-PAC File-BOM is locked
Actions (Use **AddVarToText** after each command)
Run Program to generate INI script files
open VSMS database
Query Project (**GetProject**)
open FI-project-Pac File BOM (**GetBOMData?**)
Assign Pac Files (**AutoAssignPACFiles**)
Update multiplie (set all to compressed) (**SetARCDCompressed?**)
open Program to generate INI script files
Generate ARCD recovery media Scripts (**GenerateARCDScripts**)
Select Drive to generate files to
View Scripts (Optional)
Check-in INI configuration files (**CheckIn**)
Upload to VSMS database (**UploadFiles**)
Send Release Mail for INI (**DumpText**)
Subject=VAIO INI FILES RELEASE NOTIFICATION %project name%
%phase%
Project
PC Model
Build
INI File name and unique identifier
list changes from last build
Post-conditions Tested during PAC File Creation process
Includes Check-In
Upload
Extends
Generalizes

[0042] 삭제

[0043] 삭제

[0044] 삭제

[0045] 삭제

[0046] 삭제

[0047] 삭제

[0048] 삭제

[0049] 삭제

[0050] 삭제

[0051] 삭제

[0052] 삭제

[0053] 삭제

[0054] 삭제

[0055] 삭제

[0056] 삭제

[0057] 삭제

[0058] 삭제

[0059] 삭제

[0060] 삭제

[0061] 삭제

[0062] 삭제

[0063] 삭제

[0064] 삭제

[0065] 삭제

[0066] 삭제

[0067] 삭제

[0068] 삭제

[0069] 삭제

[0070] 삭제

[0071] 삭제

[0072] 삭제

[0073] 삭제

[0074] 1.2 PAC 파일 생성(패키징된 소프트웨어)

Description Creates PAC file(s) for software recovery tools
Use Case identifier B2
Author
Date 5/01/2003
Revised
Actors Release Engineer
Pre-conditions INI file(s) created
Actions Copy files to local drive
Open browser
Browse to ARCD Scripts directory
Execute program to copy individual software locally from the network (**ExecuteProgram**)
Verify files are copied to local drive
Execute program to package each directory (**ExecuteProgram**)
Check-in PAC File(s) (**CheckIn**)
Upload to VSMS database (**UploadFiles**)
Send Release Mail for PAC File(s) (**DumpText**)
Subject=VAIO PAC FILES RELEASE NOTIFICATION %project name% %phase%
Project
PC Model
Phase
DMI information
PAC Files
PAC File Names
Changes from Last Build
Known Issues
Special Notes
Post-conditions Must be tested during software download and recovery process
Includes Create INI
Check-In
Upload PAC File(s)
Extends Create-INI
Generalizes

[0075] 삭제

[0076] 삭제

[0077] 삭제

[0078] 삭제

[0079] 삭제

[0080] 삭제

[0081] 삭제

[0082] 삭제

[0083] 삭제

[0084] 삭제

[0085] 삭제

[0086] 삭제

[0087] 삭제

[0088] 삭제

[0089] 삭제

[0090] 삭제

[0091] 삭제

[0092] 삭제

[0093] 삭제

[0094] 삭제

[0095] 삭제

[0096] 삭제

[0097] 삭제

[0098] 삭제

[0099] 삭제

[0100] 삭제

[0101] 삭제

[0102] 삭제

[0103] 삭제

[0104] 삭제

[0105] 삭제

[0106] 삭제

[0107] 삭제

[0108] 1.3 RDVD 리커버리 미디어 생성

Description Creates RDVD(s) for HDD Recovery machines that have DVD drives
Use Case identifier B5
Author
Date 5/02/2003
Revised
Actors Release Engineer
Pre-conditions Pac File(s), INI File(s), and Image File(s) are created
Actions Create PAC File(s)
Create Recovery Partition
Test Recovery Functionality
Copy files to local drive
Copy P1 Contents Local
Copy Foundation Image files(s) local
Delete the Minint Folder
Copy RDVD Boot files to Local
Create ISO File(s)
Create master RDVD(s)
Test
Check-in RDVD(s)
Turn-in RDVD(s) to Software Librarian
Send Release Mail for RDVD
Subject=VAIO RDVD FILES RELEASE NOTIFICATION %project name%
%phase%
Project
PC Model
Phase
Image Unique identifier
RDVD Unique identifier
Recovery partition Unique identifier
DMI information
Version
Media
Volume Labels
Changes from Last Build
Known Issues
Special Notes
Post ISO File(s)
Post-conditions Must be tested with the correct machine(s), DMI information
Includes Check-In
Post ISO (not created yet)
Extends None
Generalizes None

[0109] 삭제

[0110] 삭제

[0111] 삭제

[0112] 삭제

[0113] 삭제

[0114] 삭제

[0115] 삭제

[0116] 삭제

[0117] 삭제

[0118] 삭제

[0119] 삭제

[0120] 삭제

[0121] 삭제

[0122] 삭제

[0123] 삭제

[0124] 삭제

[0125] 삭제

[0126] 삭제

[0127] 삭제

[0128] 삭제

[0129] 삭제

[0130] 삭제

[0131] 삭제

[0132] 삭제

[0133] 삭제

[0134] 삭제

[0135] 삭제

[0136] 삭제

[0137] 삭제

[0138] 삭제

[0139] 삭제

[0140] 삭제

[0141] 삭제

[0142] 삭제

[0143] 삭제

[0144] 삭제

[0145] 삭제

[0146] 삭제

[0147] 삭제

[0148] 삭제

[0149] 삭제

[0150] 삭제

[0151] 1.4 HRCD 리커버리 미디어 생성

Description Creates HRCD(s) for HDD Recovery machines that do not have DVD drives
Use Case identifier B6
Author
Date 5/02/2003
Revised
Actors Release Engineer
Pre-conditions Pac File(s), INI File(s), and Image File(s) are created
Actions Create PAC File(s)
Create Recovery Partition
Test Recovery Functionality
Create master HRCD(s)
Create ISO File(s)
Test
Check-in HRCD(s)
Turn-in HRCD(s) to Software Librarian
Send Release Mail for HRCD
Subject=VAIO HRCD FILES RELEASE NOTIFICATION %project name% %phase%
Project
PC Model
Phase
Image Unique identifier
HRCD Unique identifier
Recovery Partition Unique identifier
DMI information
Version
Media
Volume Labels
Changes from Last Build
Known Issues
Special Notes
Post ISO File(s)
Post-conditions Must be tested with the correct machine(s), DMI information
Includes Check-In
Post ISO (not created yet)
Extends None
Generalizes None

[0152] 삭제

[0153] 삭제

[0154] 삭제

[0155] 삭제

[0156] 삭제

[0157] 삭제

[0158] 삭제

[0159] 삭제

[0160] 삭제

[0161] 삭제

[0162] 삭제

[0163] 삭제

[0164] 삭제

[0165] 삭제

[0166] 삭제

[0167] 삭제

[0168] 삭제

[0169] 삭제

[0170] 삭제

[0171] 삭제

[0172] 삭제

[0173] 삭제

[0174] 삭제

[0175] 삭제

[0176] 삭제

[0177] 삭제

[0178] 삭제

[0179] 삭제

[0180] 삭제

[0181] 삭제

[0182] 삭제

[0183] 삭제

[0184] 삭제

[0185] 삭제

[0186] 삭제

[0187] 삭제

[0188] 삭제

[0189] 1.5 체크인

Description Check in any item into VSMS database
Use Case identifier S1
Author
Date 5/02/2003
Revised
Actors Release Engineer
Pre-conditions None
Actions Check-in an item
Open VSMS database
Select Software Release/Submit
Select Vendor
Select Component/Release Name
Click Submit
Fill in the form completely with all applicable data
Click Submit
Post-conditions None
Includes None
Extends None
Generalizes None

[0190] 삭제

[0191] 삭제

[0192] 삭제

[0193] 삭제

[0194] 삭제

[0195] 삭제

[0196] 삭제

[0197] 삭제

[0198] 삭제

[0199] 삭제

[0200] 삭제

[0201] 삭제

[0202] 삭제

[0203] 삭제

[0204] 삭제

[0205] 삭제

[0206] 삭제

[0207] 삭제

[0208] 삭제

[0209] 1.6 VSMS 데이터 베이스로의 업로드

Description Upload an item to the appropriate locations
Use Case identifier S2
Author
Date 5/02/2003
Revised
Actors Release Engineer
Pre-conditions Item is checked in to VSMS database
Actions Open VSMS database
Select Software Release/Query
Select Vendor
Select Component/Release Name
Click on the Unique identifier for the Item
Select view item
Click on Upload
Follow on screen prompts
Post-conditions None
Includes None
Extends None
Generalizes None

[0210] 삭제

[0211] 삭제

[0212] 삭제

[0213] 삭제

[0214] 삭제

[0215] 삭제

[0216] 삭제

[0217] 삭제

[0218] 삭제

[0219] 삭제

[0220] 삭제

[0221] 삭제

[0222] 삭제

[0223] 삭제

[0224] 삭제

[0225] 삭제

[0226] 삭제

[0227] 삭제

[0228] 삭제

[0229] 1.7 ISO 파일 업로드

Description Upload an item to the appropriate locations
Use Case identifier S2
Author
Date 5/02/2003
Revised
Actors Release Engineer
Pre-conditions None
Actions Check-in an item
Open VSMS database
Select Software Release/Query
Select Vendor
Select Component/Release Name
Click Submit
Post-conditions None
Includes None
Extends None
Generalizes None

[0230] 삭제

[0231] 삭제

[0232] 삭제

[0233] 삭제

[0234] 삭제

[0235] 삭제

[0236] 삭제

[0237] 삭제

[0238] 삭제

[0239] 삭제

[0240] 삭제

[0241] 삭제

[0242] 삭제

[0243] 삭제

[0244] 삭제

[0245] 삭제

[0246] 삭제

[0247] 여기서 상세히 도시되고 기술된 특별한 소프트웨어 스위트 구축 시스템 및 방법은 전술한 본 발명의 목적을 완전하게 달성할 수 있지만, 이는 본 발명의 현재의 바람직한 실시예이므로 본 발명에 의해 광범위하게 숙고된 특허 대상을 대표한다는 점과, 본 발명의 범주가 당업자에게 명백할 수 있는 다른 실시예를 완전히 포함한다는 점과, 따라서 첨부된 청구범위 외의 어느 것에 의해서 한정되는 것이 아니며, 아울러 여기서 단수인 엘리먼트에 대한 참조는, 명백히 "하나 이상"이라고 언급되는 경우를 제외하고는, "하나 및 단 하나"를 의미하고자 한 것은 아니라는 점도 이해되어야 할 것이다. 장치 및 방법은, 본 클레임에 내포되어 있기 때문에, 본 발명이 해결하고자 하는 각각의 및 모든 문제에 대해 반드시 언급할 필요는 없다. 또한, 본 발명의 어떤 엘리먼트, 컴포넌트, 또는 방법 단계는, 상기 엘리먼트, 컴포넌트, 또는 방법 단계가 청구 범위 내에 명백히 언급되는지와는 상관없이, 대중에게 전용되도록 의도된 것은 아니다. 여기서 부재하는 표현 정의, 클레임 용어들은 본 명세서 및 파일 이력과 대립하지 않는 모든 통상의 그리고 평상의 의미로 제공된다.

도면의 간단한 설명

[0011] 도 1은 본 시스템의 블럭 다이아그램이다.

[0012] 도 2는 소프트웨어 아키텍처의 블럭 다이아그램이다.

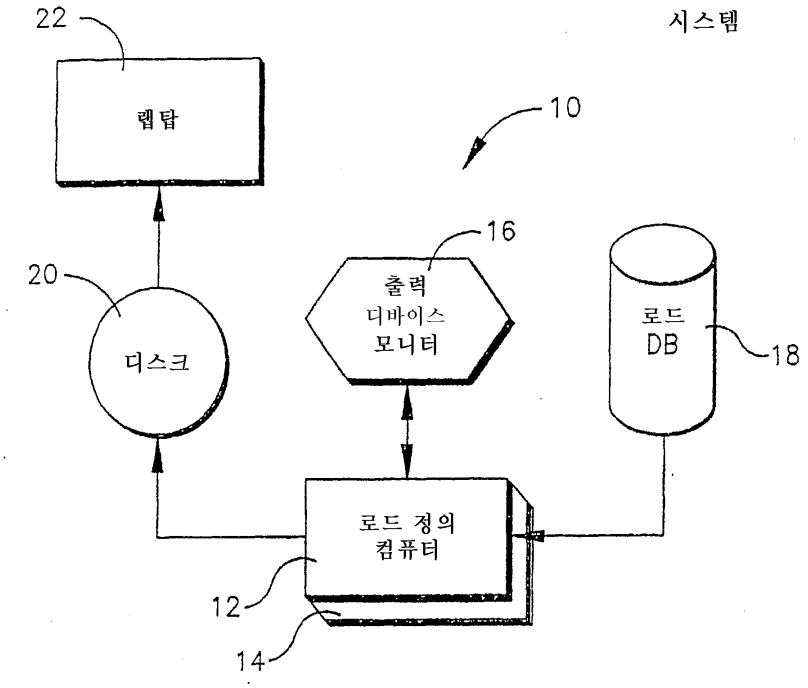
[0013] 도 3은 본 발명의 일반적 로직의 플로우 차트이다

[0014] 도 4는 XML 스크립트의 계층적 다이아그램을 나타내는 개략적 다이아그램이다.

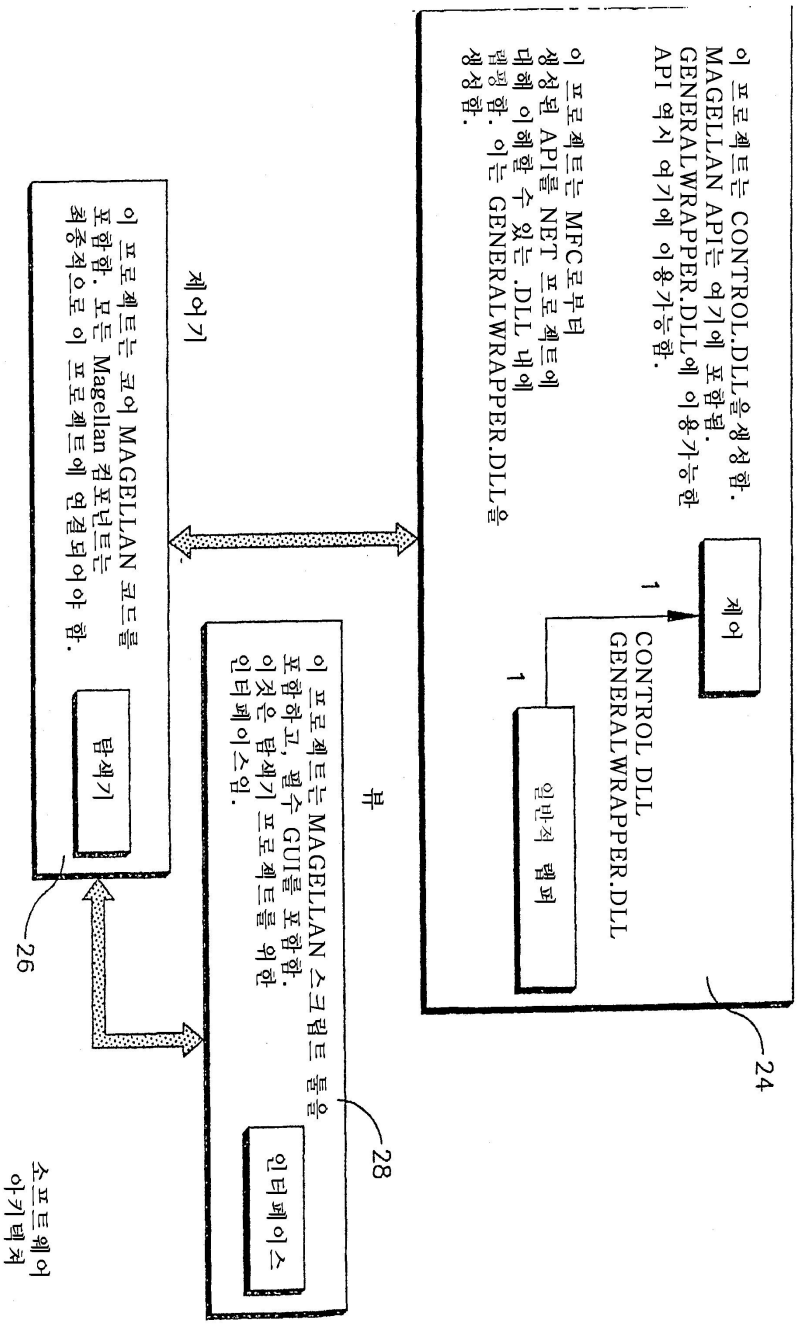
[0015] 도 5는 사용자 디스플레이를 나타내는 스크린 샷이다.

도면

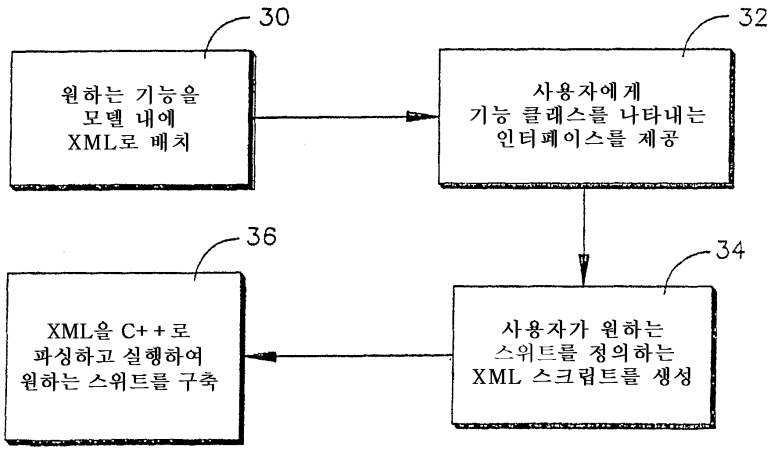
도면1



도면2

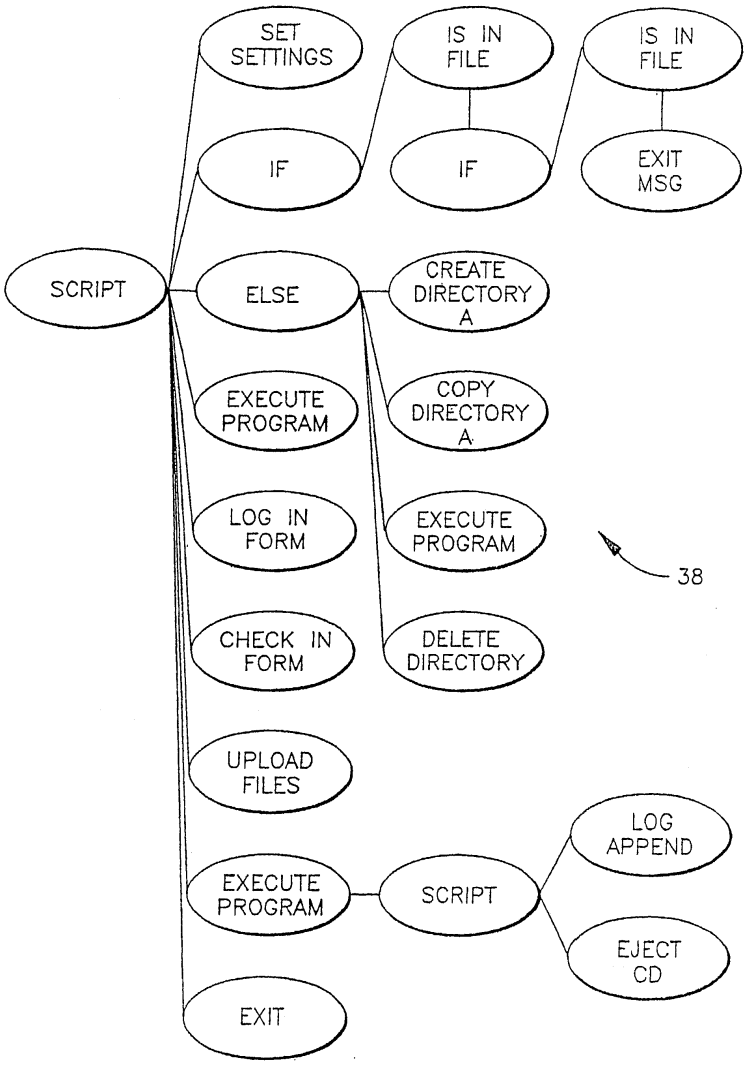

모델
이 프로젝트는 CONTROL.DLL을생성함. MAGELLAN API는 여기에 포함됨. GENERALWRAPPER.DLL에 이용가능한 API 역시 여기에 이용가능함.
제어
1
CONTROL DLL
GENERALWRAPPER.DLL
1
일반적 랩퍼
이 프로젝트는 MFC로부터 생성된 API를 NET 프로젝트에 대해 이해할 수 있는 .DLL 내에 랩핑 함. 이는 GENERALWRAPPER.DLL을 생성함.
24
제어기
이 프로젝트는 코어 MAGELLAN 코드를 포함함. 모든 Magellan 컴포넌트는 최종적으로 이 프로젝트에 연결되어야 함.
탐색기
26
뷰
이 프로젝트는 MAGELLAN 스크립트 툴을 포함하고, 몇몇 GUI를 포함함. 이것은 탐색기 프로젝트를 위한 인터페이스임.
인터페이스
28
소프트웨어 아키텍처

도면3


30
원하는 기능을
모델 내에
XML로 배치
32
사용자에게
기능 클래스를 나타내는
인터페이스를 제공
34
사용자가 원하는
스위트를 정의하는
XML 스크립트를 생성
36
XML을 C++로
파싱하고 실행하여
원하는 스위트를 구축

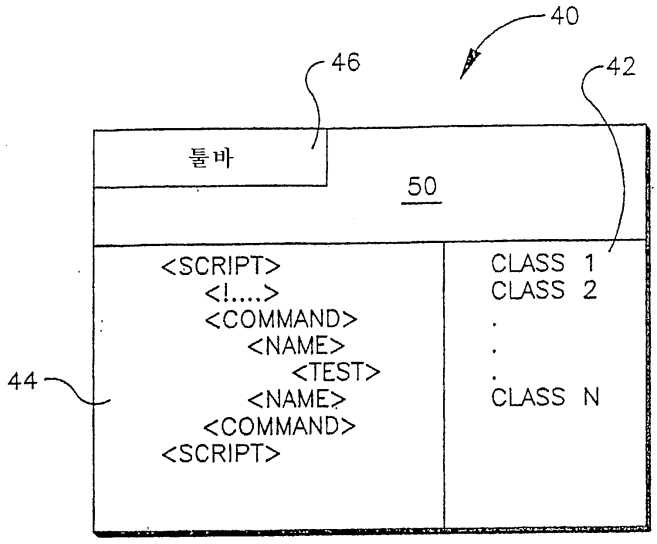
전체 로직

도면4


SET SETTINGS
IS IN FILE
IS IN FILE
IF
IF
EXIT MSG
SCRIPT
ELSE
CREATE DIRECTORY A
EXECUTE PROGRAM
COPY DIRECTORY A
LOG IN FORM
EXECUTE PROGRAM
38
CHECK IN FORM
DELETE DIRECTORY
UPLOAD FILES
LOG APPEND
EXECUTE PROGRAM
SCRIPT
EJECT CD
EXIT

스크립트

도면5


40
46
42
툴바
50
<SCRIPT>
<!....>
<COMMAND>
<NAME>
<TEST>
<NAME>
<COMMAND>
<SCRIPT>
44
CLASS 1
CLASS 2
.
.
.
CLASS N

디스플레이