



(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。 Int. Cl.

G10L 15/22 (2006.01)

G10L 15/00 (2006.01)

H04M 3/493 (2006.01)

(11) 공개번호 10-2006-0120004

(43) 공개일자 2006년11월24일

(21) 출원번호 10-2006-7006558

(22) 출원일자 2006년04월05일

심사청구일자 없음

번역문 제출일자 2006년04월05일

(86) 국제출원번호 PCT/IB2004/051995

(87) 국제공개번호 WO 2005/036526

국제출원일자 2004년10월06일

국제공개일자 2005년04월21일

(30) 우선권주장 03103755.9 2003년10월10일 유럽특허청(EPO)(EP)

(71) 출원인 코닌클리케 필립스 일렉트로닉스 엔.브이.
네델란드왕국, 아인드호펜, 그로네보르스베그 1

(72) 발명자 오에르데르, 마르틴
독일, 52066 아헨, 바이스하우스스트라췌 2, 필립스 인텔렉추얼프로퍼
티 앤드 스탠다즈 게엠베하 내

(74) 대리인 문경진

전체 청구항 수 : 총 11 항

(54) 대화 시스템을 위한 대화 제어 장치

(57) 요약

대화 시스템을 위한 대화 제어 장치(DS)를 개발하기 위한 방법과 시스템이 설명되는데, 이러한 시스템은 한편으로는 사용자와의 대화를 제어하는 작업을 가지고, 다른 한편으로는 음성 사용자 인터페이스(SP)와 대화 시스템의 애플리케이션(AP)을 감시한다. 우선 그래픽 대화 설명(GB)이 만들어지고, 그동안 개발 과정이 개발 시스템의 디스플레이 디바이스에 의해 디스플레이된다. 계속해서, 그래픽 대화 설명(GB)이 기술적 대화 설명(TB)으로 변환되고, 이러한 기술적 대화 설명(TB)은 객체 지향 번역기 언어의 클래스를 포함하고, 이들을 2진 형식으로 번역하며, 이러한 2진 형식은 궁극적으로 대화 시스템에 의해 실행될 수 있는 대화 제어 장치(DS)를 나타낸다.

대표도

도 1

특허청구의 범위

청구항 1.

음성 사용자 인터페이스(SP)와 애플리케이션(AP)을 구비한 대화 시스템을 위한 대화 제어 장치(DS)의 제작 방법으로서, 상기 대화 제어 장치(DS)는 음성 입력 인터페이스(SP) 및 애플리케이션(AP)과 함께 작용하고,

- 그래픽 대화 성분을 포함하는 그래픽 대화 설명(GB)의 제작 단계로서, 상기 그래픽 대화 설명(GB) 제작 단계 동안 상기 그래픽 대화 성분이 디스플레이 디바이스에 의해 디스플레이되는, 제작 단계,
- 상기 그래픽 대화 설명(GB)의 그래픽 대화 성분을 객체 지향 번역기 언어의 클래스 형태로 기술적 대화 설명(TB)의 기술적 대화 성분으로 변환하는 단계, 및
- 대화 제어 장치(DS)를 형성하면서, 상기 기술적 대화 설명(TB)을 상기 대화 시스템에 의해 직접 사용될 수 있는 2진 데이터로 번역하는 단계를

포함하는, 대화 시스템을 위한 대화 제어 장치(DS)의 제작 방법.

청구항 2.

제 1항에 있어서, 상기 기술적 대화 설명(TB)은 텍스트가(textually) 변경될 수 있고, 필요하다면 그래픽 대화 설명(GB)이 상기 기술적 대화 설명(TB)에 텍스트 변경에 의해 적어도 부분적으로 보충되며, 상기 보충된 그래픽 대화 설명(GB)이 디스플레이되고, 기술적인 대화 설명(TB)으로의 그래픽 대화 설명(GB)의 변환시, 기술적 대화 설명(TB)의 텍스트 변경이 고려되는 것을 특징으로 하는, 대화 시스템을 위한 대화 제어 장치(DS)의 제작 방법.

청구항 3.

제 1항 또는 제 2항에 있어서, 기존의 기술적 대화 설명(TB)은, 새로운 그래픽 대화 설명(GB)의 제작을 위한 기초로서 그래픽 대화 설명(GB)으로 적어도 부분적으로 변환되는 것을 특징으로 하는, 대화 시스템을 위한 대화 제어 장치(DS)의 제작 방법.

청구항 4.

제 1항 내지 제 3항 중 어느 한 항에 있어서, 대화 설명 클래스 계층 구조 및/또는 대화 설명 클래스 라이브러리의 주어진 객체 지향 클래스는 그래픽 대화 성분으로서 적어도 부분적으로 표현되고,

- 상기 그래픽 대화 설명(GB)은 그래픽 선택, 그래픽 대화 성분의 결합 및/또는 변경을 통해 제작되며,
- 상기 그래픽 대화 설명(GB)은 유도된 클래스의 형태로 된 기술적 대화 설명(TB)의 기술적 대화 성분으로 그래픽 대화 성분에 의해 표현되는 특정 클래스의 유도를 통해 변환되는 것을 특징으로 하는, 대화 시스템을 위한 대화 제어 장치(DS)의 제작 방법.

청구항 5.

제 1항 내지 제 4항 중 어느 한 항에 있어서, 상기 특정된 하위 대화 설명은 그래픽 대화 성분으로서 적어도 부분적으로 그래픽으로 표현되며 그래픽 대화 설명(GB)으로 선택되고 통합되는 것을 특징으로 하는, 대화 시스템을 위한 대화 제어 장치(DS)의 제작 방법.

청구항 6.

제 1항 내지 제 5항 중 어느 한 항에 있어서, 상기 그래픽 대화 설명(GB)의 경우, 모든 그래픽 대화 성분은 기호로 표현되고, 상기 그래픽 대화 설명(GB)의 제작은 상기 기호의 선택, 복사 및 링크와, 상기 기호에 의해 표현된 기술적 대화 성분의 변경을 통해 일어나는 것을 특징으로 하는, 대화 시스템을 위한 대화 제어 장치(DS)의 제작 방법.

청구항 7.

음성 입력 인터페이스(SP)와 애플리케이션(AP)을 구비한 대화 시스템을 위한 대화 제어 장치(DS)의 제작 시스템으로서, 상기 대화 제어 장치(DS)는 음성 입력 인터페이스(SP) 및 애플리케이션(AP)과 함께 작용하고,

- 그래픽 대화 설명(GB)의 그래픽 대화 성분을 시각화하고 변경하기 위한 그래픽 대화 편집기(GE),
- 객체 지향 번역기 언어의 클래스 형태로 된 기술적 대화 설명(TB)의 기술적 대화 성분으로 상기 그래픽 대화 성분을 변환하기 위한 컨버터(UM) 및
- 상기 대화 시스템에 의해 직접 사용될 수 있는 2진 데이터로 상기 대화 제어 장치(DS)를 형성하면서, 상기 기술적 대화 설명을 번역하기 위한 번역기(UB)를 포함하는, 대화 제어 장치(DS)의 제작 시스템.

청구항 8.

제 7항에 있어서, 상기 기술적 대화 설명(TB)을 변경하기 위한 텍스트 대화 편집기(TE)와, 상기 기술적 대화 설명(TB)의 텍스트 변경에 의해 상기 그래픽 대화 설명(GB)의 적어도 부분적인 보충을 위한 갱신기(AK)를 특징으로 하는, 대화 제어 장치(DS)의 제작 시스템.

청구항 9.

제 7항 또는 제 8항에 있어서, 상기 대화 제어 장치(DS)와 동일한 실행 시간(run-time) 환경에서 사용 가능한 것을 특징으로 하는, 대화 제어 장치(DS)의 제작 시스템.

청구항 10.

대화 제어 장치(DS), 음성 사용자 인터페이스(SP) 및 애플리케이션(AP)을 구비한 대화 시스템으로서, 상기 대화 제어 장치(DS)는 음성 입력 인터페이스(SP)와 상기 애플리케이션(AP)과 함께 작용하고, 상기 대화 제어 장치(DS)는 제 1항 내지 제 6항 중 어느 한 항에 따른 방법으로 제작된, 대화 시스템.

청구항 11.

제 10항에 있어서, 작동 중에 새로운 대화 제어 장치(DS)가 통합되고 배치될 수 있는 것을 특징으로 하는, 대화 시스템.

명세서

기술분야

본 발명은 대화 시스템에 관한 대화 제어 장치를 만들기 위한 방법 및 시스템과, 이러한 유형의 대화 제어 장치를 사용하는 대화 시스템에 관한 것이다.

배경기술

대화 시스템의 음성 제어 사용자 안내를 위한 대화 제어 장치는, 예를 들어 음성 제어된 정보와, 텔레뱅킹 및 홈(home) 대화 시스템과 같은 서비스 제공 시스템과 같은 모든 종류의 음성 포털(portal)에 관한 광범위한 상용 애플리케이션분야를 가진다. 이러한 유형의 음성 대화 시스템은 음성 입력 인터페이스를 가지는데, 이러한 인터페이스를 거쳐 사용자의 음성 발언이 기록되고 평가되며, 또한 일반적으로 음성 생성과 음성 출력을 위한 디바이스를 가진다. 중앙 제어 구성요소로서, 대화 시스템은 대화 제어 장치를 가진다. 이러한 대화 제어 장치는 시스템과 사용자 사이의 소정의, 상호 조건적인, 상호 하위 대화(reciprocal sub-dialog)의 형태로 된 모든 유효한 음성 대화뿐만 아니라 그것으로부터 유도될 수 있는 시스템측 응답을 구현한다. 유효한 대화의 네트워크 및 그것들의 조건부 의존도는, 대화 시스템이 상응하는 복잡한 사양(specification)을 구비하는 경우 상대적으로 광범위하게 확장될 수 있다.

원칙적으로 그러한 대화 제어 장치는 하드웨어, 예를 들어 ROM 칩의 형태로 실현될 수 있다. 그러한 하드웨어 해결책은 실행 속도의 관점에서는 소프트웨어 해결책에 비해 우수하지만, 변경된 조건에 대한 적응 가능성은 제공하지 않는다. 그러므로 대화 제어 장치를 소프트웨어로서 실현하고, 그것을 실행을 위한 대화 시스템에 이용 가능하게 하는 것은 적절한 것이다.

음성 제어된 대화 시스템의 개발과 제작에서, 제어된 대화 시스템이 상응하는 복잡성을 가지는 경우 한편으로 대화 개발자가 대화 제어 장치의 개관을 얻는 것이 어렵다는 일반적인 문제점이 생긴다. 다른 한편으로는, 대화 시스템의 사양에 대한 각각의 변경, 소위 대화 서술을 가지고, 대화 제어 장치 또는 대화 서술의 대응하는 적응이 반드시 일어나야 하고, 그러한 적응의 빠르고 정확한 실행 가능성은 대화 서술의 복잡도에 의해 차례로 제한된다. 대화 시스템의 유형이나 애플리케이션에 따라, 예를 들어 자동 판매 시스템으로부터 제공중인 재화나 용역 또는 그것들의 가격의 범위에 있어서의 변경 결과로서의 그러한 변경은 매우 자주 수행될 필요가 있다.

통상, 특별한 개발 시스템을 구비한 대화 제어 장치는 대화 개발자에 의해 만들어지고, 계속해서 대화 시스템으로 로딩되어 그것의 실행시간(run-time) 시스템에 의해 실행된다. 만약 대화 설명의 변경이 필수적이라면, 대화 제어 장치가 개발 시스템 내로 판독되어야 하고, 그에 따라 그곳에서 갱신되며 계속해서 대화 시스템의 실행시간 환경으로 다시 로딩된다. 요즘에는 대부분 소유자 설명(proprietary description) 및 규격 언어가 사용되는데, 이들은 대부분 작업에 특별히 맞게 된 명령어의 범위를 가진 프로그래밍 언어이다. 이들 언어는 대화 개발자가 대화 제어 장치를 명시할 목적으로 특별히 그 언어를 학습해야 한다는 단점을 가진다. 마찬가지로, 대화 시스템 제작자의 클라이언트나 외부 서비스 제공자는 숙달 훈련 후에만 대화 설명에 대한 적응 또는 변경을 할 수 있다. 이는 그러한 시스템을 매우 유연성이 없게 만들고, 일부 응용 분야에 있어서는 사용할 수 없게 만든다. 게다가, 만들어지거나 변경될 대화 설명은 종종 너무 복잡해서, 개발 과정이나 유지 보수 과정이 소유자 언어의 주어진 프레임워크에서 상당한 양의 시간을 차지할 수 있다.

대화 제어 장치를 위한 그러한 규격(specification) 언어는 대부분 인터프리터에 의한 대화 시스템 상에서 실행되는 스크립트(script) 언어이다. 이는 인터프리테이션(interpretation)으로 인해, 대화 시스템의 실행시간 환경에서 그러한 대화 제어 장치의 실행 속도가 제한된다는 주된 단점을 가진다. 마찬가지로, 특정 레벨의 복잡도에 도달하는 스크립트 언어 프로그램의 개발 시간은 구성하기가 더 어렵다는 사실로 인해 더 길다. 현재의 기술 개발로 인해, 명령어의 범위를 확장하거나 규격 언어의 강력함을 확장하는 것이 필수적이 될 수 있다. 이는 개발 시스템의 인터프리터와 대화 시스템의 인터프리터 모두가 새로운 범위의 명령어에 적응해야 한다는 추가 단점을 유도한다.

원칙적으로, 그래픽이나 시각적 프로그래밍 수단으로 대화 제어 장치의 입안을 실현하는 것이 또한 가능하다. 하지만, 이 경우 광범위한 대화 설명의 복잡도를 마스터하는데 있어 상당한 어려움이 발생할 수 있는데, 이는 일반적으로 그래픽 도구가 특별히 맞추어진, 따라서 그로 인한 특별히 맞추어진, 따라서 제한된 범위를 가진 명령어 구조와 감시 구조를 갖는 소유자 방법만을 제공하기 때문이다. 이러한 식으로 개발된 그래픽 개발 대화는, 이후 대화 제어 장치로서 스크립트-언어 프로그램의 형태로 대화 시스템에 이용 가능하게 만들어질 수 있다. 하지만, 이러한 식으로 실현될 수 있는 그래픽 개발 시스템은 임의의 원하는 복잡도의 대화를 개발하기 위한 필수적인 역량(power)을 가지지 않는다.

발명의 상세한 설명

그러므로 본 발명의 목적은, 대화 시스템을 위한 대화 제어 장치의 간단하고, 유연하며 빠르고 경제적인 새로운 제작 및 적응을 가능하게 하는 것이다.

이 목적은 대화 시스템을 위한 대화 제어 장치를 만들기 위한 방법에 의한 한 가지 것을 위해 달성되고, 이러한 시스템에서 애플리케이션 개발자는 먼저 적합한 그래픽으로 표현된 대화 성분을 선택하고 결합함으로써 그래픽 대화 설명을 만든다.

이 경우 그래픽 대화 설명은 디스플레이 디바이스에 의해 보여지게 된다. 대화 성분은, 예를 들어 통상 인사, 마지막 대화, 확인 대화 또는 정보 질문과 같은 자주 발생하고 재사용 가능한 표준 하위 대화를 나타낼 수 있다. 마찬가지로, 예를 들어 대화 제어 장치가 가정할 수 있는 상태와 같은 대화의 구조 성분 또는 상태 전이는 그래픽으로 표현될 수 있다. 대화 개발자는 이러한 것들을 기록 보관소(예를 들어, 대화 데이터베이스나 대화 라이브러리)로부터 취할 수 있고, 양호하게 한정된 인터페이스를 거쳐 그래픽 대화 설명의 시각화에 기초하여 그러한 것들을 이미 선택된 성분에 적절히 링크할 수 있다. 본 발명에 따르면, 이러한 그래픽 대화 설명은 기술 대화 설명으로 연속해서 자동으로 변환되고, 이러한 기술 대화 설명은 객체 지향 번역기 언어의 프로그램 코드의 형태로 생성된 대화를 표현한다. 그러한 범용(universal) 프로그래밍 언어(예를 들어, JAVA, C++ 또는 C#)는 일반적으로 그래픽 설명 언어보다 계산 측면에서 더 강력하므로, 그러한 그래픽 대화 설명의 기능은 모든 경우에 있어 변환을 통해 동등한 기술 대화 설명으로 매핑될 수 있다. 개발자가 그가 원하는 대로 완전히 기술 레벨에 대한 대화를 만든 후에는, 기술 대화 설명이 대화 제어 장치를 형성하는 적합한 컴파일러를 사용하여 기계 코드로 번역되고, 즉각적인 사용을 위해 대화 시스템에 설치될 수 있다. 인터프리터(interpreter) 언어나 스크립트(script) 언어에서 만들어진 대화 제어 장치의 일상적인 사용에 비해, 이는 대화 제어 장치가 기계 코드에서 이용 가능하다는 장점을 가지고, 이러한 점에서 대화시스템에서 빠르고 믿을 수 있게 작동하는데 반해, 스크립트 프로그램은 더 느린 인터프리터를 통해 해석되어야 한다.

소유자 프로그래밍 언어보다 범용 언어를 사용하는 것의 상당한 장점은 임의의 생각할 수 있는 대화를 실현할 수 있는 가능성에 있다. 이는 그러한 대화 제어 장치가 지출없이 임의의 시간에 확장될 수 있고, 임의의 기술 개발 또는 시장-경제 개발에 적용할 수 있다는 추가 장점을 만든다.

기술 대화 설명은 사용된 객체 지향 프로그래밍 언어의 클래스를 포함하고, 이는 한편으로는 기본 클래스로부터의 파생물이며, 따라서 이들 기본 클래스의 일정한 특징을 물려받고, 다른 한편으로는 하위 대화, 대화 상태 또는 대화 전이를 위한 하위 클래스를 포함할 수 있다. 이러한 식으로, 전체적으로 기술 대화 설명을 표현하고 궁극적으로는 대화 제어 장치를 완전히 명시하는 클래스 계층 구조가 발생할 수 있다.

임의의 원하는 객체 지향 프로그래밍 언어로 쓰여진 대화 성분이나 대화 모듈은, 언어의 구조와, 인터페이스의 정확한 한정에 의해 생긴 데이터 캡슐화(encapsulation)로 인한 장점을 가지고, 원칙적으로 재사용 가능하며, 따라서 대화 제어 장치의 연속적인 새로운 개발을 위해 효율적으로 다시 사용될 수 있다.

그래픽 선택과 결합을 통한 그래픽 레벨에 대한 대화를 개발한 대화 개발자는, 바람직하게 프로그래밍을 통한 기술 레벨에 대한 대화를 좀더 정확하게 실행하고 세련되게(refine) 할 수 있다. 그러므로, 예를 들어 기술 설명 레벨에 대한 대화 시스템의 표준 발언은 구체적인 대화 상황에 적응될 수 있거나, (그래픽) 하위 대화에서 명시되는 사용자에게 개방된 질문 옵션이 다른 사람들을 추가함으로써 확장될 수 있다.

기술 설명 레벨에 대한 대화 클래스의 새로운 기입은, 개발자들이 그것을 그래픽으로 개발할 수 있는 것보다 빠르게 대화를 프로그래밍할 수 있다면, 특히 숙련된 대화 개발자에게 유리하게 될 수 있다. 마찬가지로, 개발에 대한 기술 프로그래밍 접근은, 예를 들어 특별한 레벨의 추상화의 대화 클래스가 요구된다면 바람직할 수 있고, 이는 하나 이상의 클래스로부터의 특징과 방법을 물려받으며, 유사한 클래스는 대화 클래스에 관한 데이터베이스에서 이용 가능하지 않다. 특정 대화 상황이 실현될 수 없거나, 완전한 계산 능력을 가지지 않는 그래픽 설명 언어에 의해, 오직 어렵게 실현될 수 있는 경우에도 그러할 수 있다. 그러한 경우, 대화 개발자는 완전한 계산 능력을 가지는 프로그래밍 언어로 기술 대화 설명을 추가로 개발하고, 이를 기술 대화 설명으로 통합하기 위해 구체적인 하위 대화를 실현하는 새로운 객체 지향 클래스를 기입하는 옵션을 가진다.

어느 경우든, 대화 제어 장치의 개발은 그래픽과 동등한 프로그래밍 언어 레벨 모두에서, 특별히 유연하고 문제-지향 방식으로 진행할 수 있다는 것이 본 발명의 특별한 장점이다.

이러한 분산 대화 개발의 추가 장점은 음성 제어된 대화 시스템의 대화가 그래픽 설명 레벨에서 시각화된다는 사실에 있고, 이 경우 이들 대화는 나중 동작시 어쿠스틱 형태로만 존재하고, 오늘날 이들 대화 시스템은 상당한 정도의 복잡성에 도달할 수 있다. 그러므로 대화 개발자는 그것들에 대한 더 나은 개관을 얻을 수 있고, 그것들을 더 양호하게 설계할 수 있다. 기술 대화 설명으로 변환하는 중간 단계는 대화 개발자가 대화 제어 장치의 그래픽 설계를 대화 시스템에 의한 요구사항 세트나 클라이언트의 소망에 정확하게 적응시킬 수 있게 한다.

이 경우, 소유자 대화 설명 언어 대신 Java나 객체 지향 C-파생어와 같은 종래의 프로그래밍 언어를 사용하는 것이 특히 유리한데, 이는 대화 시스템의 제작자뿐만 아니라, 예를 들어 클라이언트의 유지보수 직원이나 전문 서비스 제공자가 대화 제어 장치를 개발할 수 있고, 필요하다면 그것을 적응시킬 수 있기 때문이다. 이 경우 필요한 것은 시스템의 프로그래밍 인터페이스(API)에서의 명령어이다. 원칙적으로, 임의의 객체 지향 프로그래밍 언어가 이 경우 선택될 수 있다.

본 발명의 일 실시예에서 그래픽 대화 설명과 기술 대화 설명 사이의 일관성을 보장하기 위해서는, 기술 대화 설명의 텍스트 변경을 그래픽 대화 설명으로 다시 통합하는 것이 가능하다. 그래픽 설명 언어와 기술 설명 언어의 강력함의 가능한 차이점은 그래픽 대화 설명의 갱신시 어떠한 문제점도 일으키지 않는데, 이는 객체 지향 프로그래밍 언어의 엄격한 구문론과 물려받음과 추상화 메커니즘으로 인한 계층적 구조를 통해, 각 대화 클래스가 그것의 의존도와 인터페이스를 가지고 그래픽으로 기호화될 수 있기 때문이다.

본 발명의 특별히 유리한 일 실시예에서, 그래픽 대화 설명의 기술 설명으로의 변환은, 기술 대화 설명의 변경에 의한 그래픽 대화 설명의 갱신과 함께, 자동화된 방식으로, 그리고 개발 환경을 통한 배경에 있는 개발자에게 투명한 방식으로 발생한다. 이를 통해, 동시에 양 설명 레벨에 대한 대화 개발자에 의한 준-병렬(quasi-parallel) 개발이 심지어 가능하다.

게다가 이러한 실시예의 경우, 그래픽 대화 설명의 기술 대화 설명으로의 사용자 제어된 변환의 경우에서 2가지 설명의 일관성은, 가능하게 변경된 클래스의 단순한 중복 기입이나 기술 대화 설명의 성분에 의해 위험에 빠지지 않지만, 그래픽 대화 설명과 비교된 기술 대화 설명의 나머지는 변환 동안 보존되고, 차례로 그래픽 대화 설명에서 갱신된다는 것이 보장된다.

기술 대화 설명의 제작을 위한 클래스/성분은 대화 데이터베이스나 라이브러리에서 전부 또는 일부, 즉 또한 개별적으로 저장될 수 있다. 일 실시예의 경우 대화 데이터베이스에 저장되는 대화 또는 하위 대화의 그러한 기술 설명은, 대화 개발 환경으로부터 유리하게 판독될 수 있고, 기술 및 -갱신 후- 대화 제어 장치의 새로운 그래픽 개발에 관한 시작점으로서 사용될 수 있다. 이는 프로그램 단편의 재사용 가능성을 가능하게 하고, 따라서 개발의 속도와 효율을 높일 수 있게 한다.

데이터베이스에서 전부 또는 부분적으로 저장되는 기술 대화 설명은, 여러 개의 추상화 레벨에서 대화의 특별히 중요한 특징을 표현하는 클래스 계층 구조와 클래스 라이브러리를 포함할 수 있다. 이들 클래스 계층 구조는 또한, 기술 대화 설명의 형태로 개발 환경으로 판독된 후에, 클래스와 물려받은 방법 사이의 관계와 특징이 적합한 그래픽 기호로 시각화되는, 그래픽 대화 설명으로서 표현될 수 있다. 이러한 식으로 시각화된 클래스 계층 구조로부터, 대화 개발자는 대응하는 기호를 선택함으로써, 그가 요구하는 것에 관해 충분히 명시되는 클래스를 선택할 수 있고, 그것들을 다른 대화 성분과 적절히 결합할 수 있다. 개별 하위 대화와 이것들 사이의 인터페이스의 정확한 규격은, 예를 들어 선택되고 결합된 성분을, 개별 클래스를 프로그래밍함으로써, 기술 대화 설명으로 변환한 후 일어날 수 있다. 이러한 식으로, 유리한 분산 대화 개발이 실현되고, 이 경우 대화 제어 장치의 구조는 명확한 그래픽 설명 레벨 상에서 실현되고, 세부 작업은 프로그래밍을 통한 기술 설명 레벨에서 일어난다.

본 발명에 따른 방법의 유리한 일 실시예의 경우, 기술 대화 설명의 모든 기술 성분/클래스는 그래픽 대화 설명에서 기호나 블록 라인 또는 원형 라인으로 표현된다. 그래픽 대화 설명은 이후 대화 제어 장치를 통해 실현될 대화의 완전한 상태/전이도로 표현될 수 있다. 이는 대화를 입안하기 위해 종종 사용되는 상태/전이도가 그래픽, 따라서 궁극적으로 모든 클래스나 성분이 이미 존재한다면 기술 대화 설명으로 직접 변환될 수 있다는 장점을 가진다.

본 발명의 이러한 실시예의 경우, 클래스의 특징 및 방법은 유리하게 기호를 통한 클래스의 요소로서 그래픽 대화 설명에서 식별 가능하게 만들어질 수 있다. 각 기호는, 예를 들어 입력 파라미터와 출력 파라미터 또는 인터페이스와 같은 추가 정보를 담고 있거나 가능하게는 이러한 방법 또는 클래스의 완전한 프로그램 코드를 심지어 기술하는 라벨을 포함할 수 있다. 마찬가지로, 개별 대화 상태 사이의 전이는 그래픽 표현되고, 이 경우 각 전이에는 시스템이나 사용자의 대응하는 진술과 대화 단계를 구비한 라벨이 할당될 수 있으며, 이는 대응하는 전이를 유도하거나 그러한 결과를 초래한다. 이러한 식으로 표현되는 그래픽 대화 설명을 가지고, 대화 개발자는 본질적으로는 언어적인 예컨대, 어쿠스틱한 대화를 그래픽 개발하고 그러한 언어적인 대화를, 끌어오기(dragging), 이동하기(moving), 복사(copying), 삽입 또는 절단과 같은 간단한 마우스 동작을 통해 좀더 정확하게 명시할 수 있다. 특별히 유리한 실시예에서, 대화 개발자는 기호를 더블 클릭함으로써, 대응하는 성분으로 텍스트 변경을 하거나 이들을 프로그래밍할 수 있는 텍스트 윈도우(window)를 열 수 있다. 이러한 종류의 그래픽 통합과 텍스트 개발을 통해, 편리한 대화 개발이 가능하게 된다.

그러한 대화 시스템에 관한 대화 제어 장치의 실시예에 있어, 본 발명에 따른 개발 시스템이 요구되고, 이는 그래픽 대화 설명이 시각화되고 편집될 수 있는 적어도 하나의 그래픽 대화 편집기를 포함한다. 대화 시스템을 위한 이러한 개발 환경

은, 또한 그래픽 대화 설명을 객체 지향 프로그래밍 언어의 클래스로 이루어지는 기술 대화 설명으로 변환하는 컨버터를 포함한다. 이러한 소스 코드는 이후 본 발명에 따른 개발 시스템의 번역기(컴파일러)에 의해 2진 설명 형식으로 번역되고, 이는 궁극적으로 대화 시스템 상에서 실행될 수 있는 대화 제어 장치를 표현한다.

기술 설명을 동등한 그래픽 대화 설명으로 역변환하기 위해서는, 개발 시스템이 갱신기(updater)를 포함하는 것이 바람직하다.

대화 제어 장치의 소스 코드를 표현하는 기술 대화 설명을 적응시키기 위해서, 개발 시스템은 소스 코드의 텍스트 편집을 위한 텍스트 편집기를 포함하는 것이 바람직하다.

본 발명의 특별히 유리한 일 실시예에서는, 텍스트 편집기 외에, 통합된 디버거(debugger), 컴파일러 및 객체 지향된 대화 클래스를 선택하기 위한 클래스 브라우저(browser)를 구비한 완전한 프로그래밍 환경이 또한 존재한다.

이러한 개발 환경의 장점은, 복잡하고 다층으로 된 어쿠스틱 대화가, 시각 프로그래밍을 통한 편집과 시각화를 위한 적당한 도구에 의해 그래픽 레벨 상에서 대화 개발자에 의해 입안되고, 동시에 개발자가 완전한 프로그래밍 환경에 의한 객체 지향 프로그래밍을 통해 기술 대화 설명으로 세부 사항들을 실현할 수 있다는 사실에 있다.

본 발명의 유리한 일 실시예로서, 설명된 개발 시스템은, 대화 시스템의 실행시간 환경으로 통합되어, 대화 설명이 대화 제어 장치로서 동작하게 되는 하드웨어 플랫폼 상에서 유리하게 만들어질 수 있게 된다.

본 명세서에서 설명된 개발 시스템으로 개발된 대화 제어 장치에 의해 제어되는 대화 시스템의 경우, 번역된 대화 제어 장치가 동작 중에 시스템으로 통합될 수 있다는 점에서 특별히 유리하다. 이를 통해, 갱신된 대화 제어 장치를 설치하기 위해서는 시스템이 동작으로부터 벗어나야 할 필요없이 대화 제어 장치가 갱신될 수 있다. 이는 예를 들어 꾸준하고 특별한 성공으로 인해, 특별히 자주 갱신될 필요가 있는 시스템의 경우에 특별히 유리한 점이다.

첨부된 도면과 이후 설명된 예시적인 실시예를 참조하여 본 발명을 상세히 설명한다.

실시예

애플리케이션(AP), 음성 생성 유닛(SG) 및 음성 입력 인터페이스(SP)와 함께 작용하는 본 발명에 따른 대화 제어 장치(DS)를 구비한 대화 시스템의 설계 예가 도 1에 도시되어 있다. 대화 제어 장치(DS)는 그것에 의해 구현된 상태, 전이 및 대화에 따라 대화 시스템을 제어한다.

들어오는 음성 발언(S)은 먼저 음성 입력 인터페이스(SP)의 신호 기록 유닛(SA)에 의해 디지털 오디오 음성 신호(AS)로 모두 변환되고, 음성 인식 유닛(SE)으로 보내진다. 음성 이해의 과정, 즉 음성 인식 유닛(SE)에 알려진 발언의 확인이 시작 신호(ST)를 통한 대화 제어 장치(DS)에 의해 개시된다.

대개, 음성 입력 인터페이스(SP)에 통합된 음성 인식 유닛(SE)은, 구문 분석 유닛과 의미 합성 유닛(본 명세서에서도 둘 다 도시되지 않음)을 포함하고, 이들은 대화 제어 장치에 의해 규정되는 사용자 문법(GR)에 따라 디지털화된 음성 발언(AS)의 유효성을 확인하고, 그것을 사용자의 발언에 대한 응답으로서 프로그래밍 언어 또는 기계 코드 명령어를 포함하는 인식 결과(ER)로 변환한다. 그러한 인식 결과(ER)는 한편으로 그 인식 결과(ER)가 대화 절차의 제어를 다시 얻는 것을 허용하기 위해 대화 제어 장치(DS)로 보내지고, 다른 한편으로는 직접 그 인식 결과(ER)에 의해 실행되도록 하기 위해 애플리케이션(AP)에 보내진다. 대안적으로, 그러한 인식 결과(ER)는 그것에 의해 애플리케이션(AP)으로 보내지도록 하기 위해 음성 인식 유닛(SE)에 의해 대화 제어 장치(DS)에만 보내질 수 있다.

그러므로 대화 제어 장치(DS)는 대화 시스템의 중심 스위칭 포인트가 되는데, 이는 그것이 음성 인식 유닛(SE)에 의해 유효한 것으로 받아들여져야 할 대화를 명시하고, 이를 통해 애플리케이션(AP)을 간접적으로 제어하기 때문이다. 게다가, 대화 제어(DS)는 또한 대화 제어 장치(DS)로 구현된 대화에 따라, 시스템에 의해 사용자로서의 음성 발언의 생성을 개시하는 음성 생성 유닛(SG)을 제어한다.

대개, 음성 입력 인터페이스(SP), 애플리케이션(AP) 및 대화 제어 장치(DS)의 음성 인식 유닛(SE)은 모두 동일한 객체 지향 번역기 언어로 기입되거나 적어도 동일한 플랫폼 상에 실행될 수 있는 언어로 기입된다.

도 2는 대화 제어 장치 개발의 개략적인 시퀀스를 도시한다. 도 2 또한 본 발명에 따른 대화 개발 환경의 모든 중요한 성분을 도시한다. 이는 대화 개발자가 대화를 시각화하고 드래프트(draft)할 수 있는 디스플레이와 편집 디바이스를 표현하는 그래픽 대화 편집기(GE)를 포함한다. 이를 위해, 예를 들어 클래스 계층 구조의 기본 클래스로서 기술적으로 실현되고, 사용자가 그래픽 대화 편집기(GE)에서의 그래픽 기호로서 선택할 수 있으며, 양호하게 한정된 인터페이스를 통해 다른 성분과 적절히 결합할 수 있는 대화, 상태 및 전이와 같은 다양한 기본 성분이 사용자에게 이용 가능하다.

이러한 제 1 개발 사이클의 결과는 편집기의 내부 표현 형태로 처음에 가상적으로만 존재하는 그래픽 대화 설명(GB)이다. 그래픽 대화 설명(GB)은, 그래픽 대화 설명(GB)의 개별(그래픽) 성분이 객체 지향 프로그래밍 언어의 클래스 인스턴스(instance)로 변환된다는 점에서 컨버터(UM)에 의해 기술적 대화 설명(TB)으로 변환된다. 이들 클래스 인스턴스는, 예를 들어 대화, 상태 및 전이의 인스턴스로서, 즉 다시 말해 특별한 하위 대화에 관해 계승된 특징을 구비한 유도된 클래스로서 프로그래밍 언어 형태로(도 4 및 도 5) 된 그래픽 성분을 표현한다.

컨버터(UM)에 의해 생성된 기술적 대화 설명(TB)은 텍스트 편집기(TE)에 의해 대화 개발자를 통해 편집될 수 있고, 이러한 텍스트 편집기(TE)는 컨버터(UM)에 의해 생성된 프로그래밍 언어에 관한 완벽한 프로그래밍 환경의 일부일 수 있다. 기술적 대화 설명(TB)의 편집 가능성을 통해, 대화 개발자는 개발될 대화 제어 장치(DS)를 프로그래밍할 수 있는데, 즉 개발자가 그래픽 대화 설명(GB)을 드래프트할 때 실현할 수 없었거나 개발자가 기술적인 프로그래밍 레벨 상에서 실현하기에 더 용이한 세부 사항으로 그것을 보충할 수 있다.

대화 개발자가 기술적 대화 편집기(TE)에 의해 기술적 대화 설명(TB)에 행하는 변경은, 대화 개발 추상화의 2개 레벨이 서로 일치하게 유지되도록, 그래픽 대화 설명으로 갱신기(AK)에 의해 재 통합될 수 있다. 이 과정에서, 새롭게 프로그래밍된 클래스/성분이나 기술적 대화 설명(TB)의 방법은 대응하는 기호에 의해 그래픽 대화 설명(GB)에서 시각화되고, 이들은 그래픽 대화 설명(GB)의 다른 기호와 새롭게 프로그래밍된 성분의 관계에 따라 링크된다.

본 발명의 특별히 유리한 일 실시예에서, 갱신(AK)은 갱신된 그래픽 대화 설명(GB)의 다시 새롭게 된 변환(UM)을 통해, 대화 개발자에 의해 변경된 본래의 기술적 대화 설명(TB)의 거동과 동일한 실행 거동을 보여주는 적절한 컴파일러에 의한 번역 후에 기술적 대화 설명(TB)이 생성된다.

대화 데이터베이스(DB)에서, 하위 대화와 표준화된 대화 소자는 프로그램 코드의 형태로 저장되고, 이는 기술 대화 설명(TB)으로서 텍스트 대화 편집기(TE)로 판독될 수 있으며, 새롭게 개발될 대화를 위한 기초로서, 또는 기존 대화의 확장/변경으로서 대화 개발자에 의해 변경될 수 있다. 마찬가지로, 기존의 대화와 대화 성분은 나중에 사용하기 위해 데이터베이스(DB)에 저장될 수 있다. 객체 지향 프로그래밍 언어가 기술 대화 설명(TB)을 위해 사용되므로, 대화 데이터베이스(DB)는 프로그래밍을 하기 위한 대화 개발자의 목적에 적절한 추상화 레벨의 클래스를 대화 개발자가 선택할 수 있는 클래스 계층 구조를 포함할 수 있다. 이들 클래스 계층 구조와 라이브러리로부터 시작하여, 새로운 대화의 개발은 유도, 도출 또는 계승 및 추상화를 통해 기술 및 그래픽 설명 레벨(TB, GB) 모두에 대해 일어날 수 있다.

완성된 기술 대화 설명(TB)은 기계 코드 대화 제어 장치(DS)로 컴파일러(UB)에 의해 번역되고, 이러한 기계 코드 대화 제어 장치(DS)는 도 1에 도시된 대화 시스템으로 최종적으로 통합된다.

대화 제어 장치(DS)의 (결정론적) 거동은 시스템의 모든 상태와 상태 변화(전이)를 초래하는 이벤트(event)를 완전히 설명하는 상태/전이도에 의해 정식으로 설명될 수 있다. 도 3은, 일 예를 통해 대화 제어 장치(DS)에 의해 실현되는 간단한 대화(HD)의 상태/전이도를 도시한다. 대화(HD)는 더 상세히 명시되지 않는 하위 대화(SD)를 가지는 상태(S1)를 가정할 수 있고, 이 경우 차례로 적어도 하나의 추가 상태가 명시되며, 또한 4개의 전이(T1, T2 및 T3/T4)를 가지고, 이들은 대화 단계에서 각각 개시되는 것이다. 전이(T1)는 상태(S1)를 자신에(on itself) 매핑하는데 반해, 다른 전이(T2, T3/T4)는 상태(S1)와 하위 대화(SD)에서 명시된 상태 중 한 상태 사이의 상태 변화를 설명한다.

상태(S1)는 대화 시스템의 초기 상태 또는 시작 상태이고, 이는 다시 사용자가 각 대화의 끝에 있음을 가정한 것이다. 이 상태에서, 시스템은 사용자가 "What can I do for you?"라는 발언을 하도록 하는 시작 문구를 생성한다. 2개의 음성 발언, 즉 전이(T1)를 개시하는 "What time is it?"과, 전이(T2)를 개시하는 "What is the weather forecast?"가 이제 사용자에게 열려 있다. 첫 번째 경우, 시작 발언을 다시 주기 위해 시작 상태(S1)로 되돌아간다는 점에서, 시스템은 올바른 시간에 응답하고 대응하는 전이(T1)를 완성시킨다. 두 번째 경우에서는 시스템이 전이(T2)와 입력 인터페이스(IN)를 거쳐 하위 대화(SD)로 바뀐다. 하위 대화(SD)에서, 전이(T3/T4)는 출력 인터페이스(OUT)를 거쳐 시작 상태(S1)로 되돌아가며 이 상태(S1)에서, 전이(T2)시 물어본 질문에 대한 답이 주어진다.

도 3b는 하위 대화(SD)의 대응하는 상태/전이도를 도시한다. 이는 입력 인터페이스(IN)와 출력 인터페이스(OUT)를 거쳐 도 3a에 도시된 더 높은 차수의 주된 대화(HD)와 링크되는 상태(S2)를 도시한다. 하위 대화(SD)는, 주된 대화(HD)의 상태에서부터의 전이(T2)와, 둘 다 주된 대화(HD)에서 명시된 상태의 변경을 시작하는 2개의 전이(T3, T4)를 거쳐 도달되는 상태(S2)를 가진다.

전이(T2)의 "What is the weather forecast?"의 대화 단계 후에는, 사용자 요청의 좀더 정확한 명시를 위해, 하위 대화(SD)는 상대 질문인 "For tomorrow or for next week?"에 응답하고, 새로운 상태(S2)로 바뀐다. 상태(S2)에서는, 사용자가 대화 단계인 "For tomorrow" 또는 "For next week"로 대화 제어 장치(DS)의 상대 질문에 대답할 수 있다. 상태(S2)에서는 사용자가 더 이상 시간을 묻는 옵션을 가지지 않고, 이를 위해 사용자는 먼저 하위 대화(SD)를 떠나야 하고, 상태(S1)가 재개되어야 한다. 사용자로부터의 설명에, 대화 제어 장치(DS)는 "for tomorrow" 또는 "for next week"의 일기 예보로 차별적으로 대답하고, 대응하는 전이(T3 또는 T4)에 의해 출력 인터페이스(OUT)를 거쳐 주된 대화(HD)로 다시 분기한 다음, 시작 상태(S1)로 되돌아 온다.

도 3a와 도 3b에서의 대화는 그래픽 대화 설명(GB)의 일 예를 표현한다. 대화 개발자는 대화(HD)를 상태(S1, S2), 전이(T1, T2, T3, T4) 및 하위 대화(SD)의 규격을 통해 그래프 형식으로 설계한다. 이 경우, 대화 단계는 더블 클릭을 하고 텍스트를 입력함으로써, 대응하는 전이(T1, T2, T3, T4)와 연관될 수 있다. 그래픽 대화 설명(GB)을 기술 대화 설명으로 변환(UM)할 때, 대화(HD)의 성분(상태, 전이 및 하위 대화)은 객체 지향 프로그래밍 언어의 클래스 형태의 프로그래밍 언어 버전으로 변환된다. 도 3a와 도 3b로부터의 그래픽 대화 설명의 그러한 변환(UM)은, 도 4와 도 5에 도시되어 있다.

설명을 위해, 다음 예시적인 기술 대화 설명(TB)이 객체 지향 프로그래밍 언어인 C#으로 기입된다. 하지만 근본적으로 임의의 다른 객체 지향 프로그래밍 언어가 기술 대화 설명(TB)의 정의에 적합하다. 도 4는 시스템에 의해 규정되는 기본 대화 클래스인 "Dialogue"로부터 유도되고 클래스 "Dialogue"의 특징을 물려받은 클래스 "ExampleDialogue"로서의 주된 대화(HD)를 도시한다. "ExampleDialogue"는 2개의 상태 "state1"과 "state2", 2개의 전이 "WeatherTrans"와 "timeTrans" 및 하위 대화 "Weather"를 정의한다. 반면에 상태 "state1"과 "state2"와 전이 "WeatherTrans"는 상태와 전이에 관한 기본 클래스인 "State"와 "Transition"의 인스턴스이고, "timeTrans"와 "Weather"는 대화 개발자에 의해 마찬가지로 구현된 2개의 클래스의 인스턴스를 형성하며, 이들은 다른 기본 클래스로부터 유도되고, 도 5에 명시되어 있다.

4명의 구성자를 부름으로써 예시할 때, 클래스 "ExampleDialogue"의 Initialize() 방법은 객체인 "state1", "WeatherTrans", "timeTrans" 및 "Weather"를 만든다. 객체 "state1"은 대화 시스템이 초기에 사용자와 통신하는 대화 단계와 연관되는 시작 상태(S1)를 표현한다. 객체 "WeatherTrans"는 상태 "state1"(S1)로부터 하위 대화(SD)의 상태인 "Weather"로 변경하는 전이(T2)를 표현하고, 그러한 식으로 주어진 문구를 출력한다. 객체 "timeTrans"는 상태인 "state1"를 자신에(onto itself) 매핑하고, 객체 "Weather"는 상태 "state1"(S1)로부터의 분기 가능한 하위 대화(SD)를 표현한다.

대화(HD)의 완전한 규격을 위해, 현재 빠진 것은 이러한 레벨 상에서 아직 알려지지 않은 클래스인 "TimeTransition"과 "WeatherDialogue"로서, 이들 인스턴스는 객체인 "timeTrans"와 "Weather"를 표현한다. 이들은 도 5에 도시되어 있다.

클래스 "TimeTransition"은 클래스 "Transition"로부터 유도되고, 이것의 객체인 주된 대화(HD)의 객체 "WeatherTrans"는 이미 한 인스턴스를 표현하는 것이다. 클래스는 질문에 대한 대답만을 실현하고, 이는 주된 대화(HD)의 클래스인 "ExampleDialogue"에서의 그것의 예시에 대한 객체인 "timeTrans"에 파라미터로서 이미 주어진 것이다.

클래스 "WeatherDialogue"는 클래스 "sub-dialogue"로부터 유도되고, 이는 차례로 상태인 "state"(S2)와 2개의 전이 "TomorrowTrans"(T3)와 "WeekTrans"(T4)를 포함한다. 이 클래스는 도 3b에 도시된 하위 대화(SD)의 기술 프로그램 변환(UM)을 표현한다. 이는 상대 질문을 가진 상태인 "state"(S2)에서의 상태/전이도에 따른 일기 예보 ("ExampleDialogue"에서의 예시 실증 파라미터와 비교)에 대한 질문에 응답하고, 이후 차별화된 방식으로 전이인 "tomorrowTrans"(T3) 또는 "WeekTrans"를 가지고 응답한다. 하위 대화를 "Exit"로 끝냄으로써, 이후 제어를 클래스 "ExampleDialogue"의 더 높은 차수의 객체로 되돌려 보내고, 이(ExampleDialogue)는 클래스 "WeatherDialogue"의 객체인 "Weather"를 예시하였다.

주된 대화(HD)와 하위 대화(SD)를 실현하기 위해 전술한 3개의 클래스를 포함하는 기술 대화 설명(TB)은 컴파일러(UB) (예, 도 2와 비교)에 의해 기계 코드로 최종적으로 번역되고, 이를 통해 대화 제어 장치(DS)를 형성한다. 대화 시스템에서 대화 제어 장치(DS)의 실행시간 동안, 클래스인 "ExampleDialogue"의 더 높은 차수의 객체가 대응하는 구성자를 부름으로써 예시되고, 이를 통해 차례로 추가 객체의 실증이 완전한 대화를 실현한다.

결론을 내리면, 도면과 상세한 설명에 의해 표현된 구체적인 대화 시스템과 개발 시스템은 단지 설계 예이고, 이는 당업자라면 본 발명의 범주를 벗어나지 않으면서 다양하게 변경시킬 수 있는 것이라는 점이 다시 한번 지적된다. 특히 본 명세서에서 도시된 설계 예에서 객체 지향 프로그래밍 언어인 C#으로 기입되는 프로그램 단편은 임의의 다른 객체 지향 프로그래밍 언어로 기입될 수 있다. 또한 본 명세서에 도시된 대화 설계 예는 매우 간단하고, 짧은 예이며, 이들은 본 발명을 가능한 간단하게 설명하기 위해 선택된 것이라는 점이 명백하다. 실제로, 대화는 당연히 상당히 더 복잡하다. 더 완벽하기 하기 위해, 단수 표현의 사용은 해당 특징이 또한 여러 번 존재할 수 있다는 가능성을 배제하지 않고, "포함한다"라는 동사의 사용은 추가 요소 또는 단계의 존재를 배제하지 않는다는 점이 또한 지적된다.

산업상 이용 가능성

전술한 바와 같이, 본 발명은 대화 시스템에 관한 대화 제어 장치를 만들기 위한 방법 및 시스템과, 이러한 유형의 대화 제어 장치를 사용하는 대화 시스템에 이용 가능하다.

도면의 간단한 설명

도 1은 대화 시스템의 개략적인 표현을 도시하는 도면.

도 2는 본 발명에 따른 대화 제어 장치의 개발의 흐름도를 도시하는 도면.

도 3a는 대화 시스템의 대화의 그래픽 표현을 도시하는 도면.

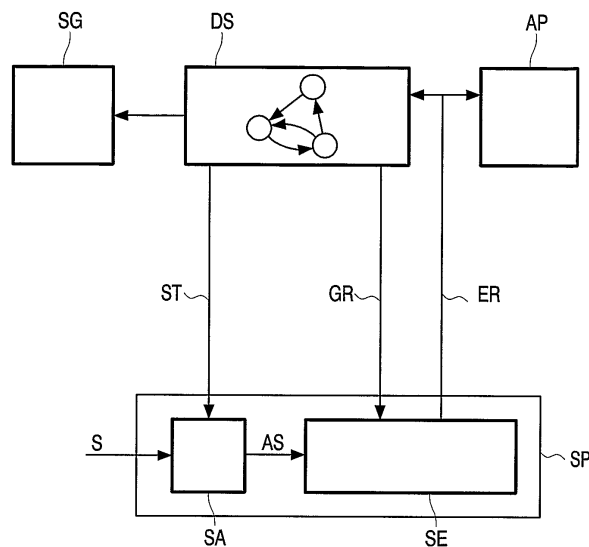
도 3b는 대화 시스템의 하위 대화의 그래픽 표현을 도시하는 도면.

도 4는 기술적인 대화 제어 장치의 일 예를 도시하는 도면.

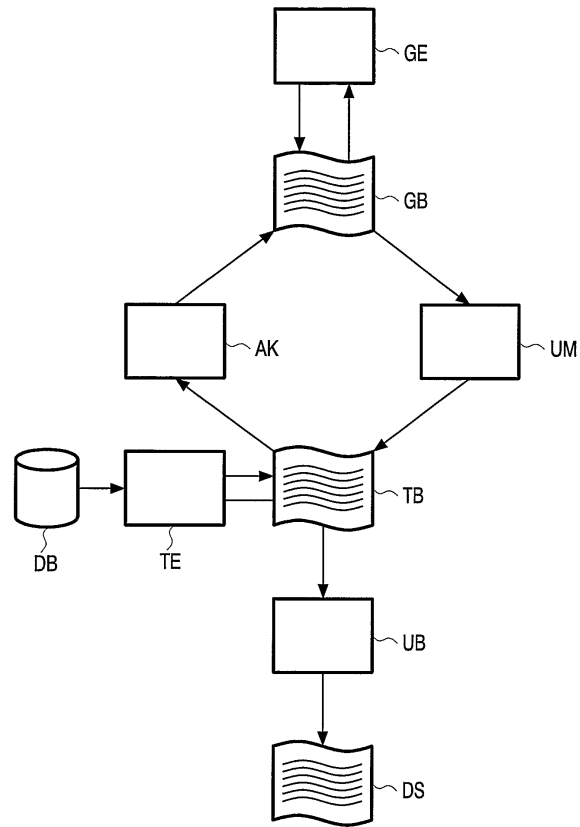
도 5는 도 4에 도시된 기술적인 대화 제어 장치의 2개의 하위 대화를 도시하는 도면.

도면

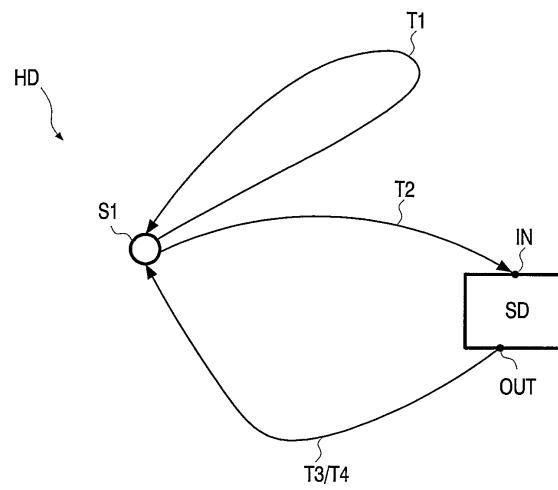
도면1



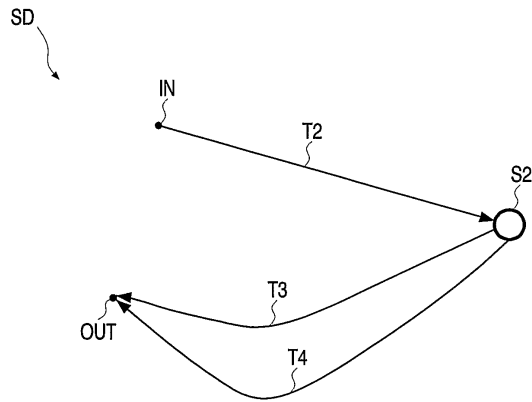
도면2



도면3a



도면3b



도면4

```

public class ExampleDialogue inherits Dialogue
{
    private State state1, state2;
    private Transation weatherTrans;
    private TimeTransition timeTrans;
    private WeatherDialogue weather;

    void Initialize ()
    {
        state1 = new State ("What can I do for you");
        weatherTrans = new Transition (state1, weather,
            "What is the weather forecast?");
        SD → timeTrans = new TimeTransition (state1, state1,
            "Tell me the time");
        SD → weather = new weatherDialogue (state1);
        StartState = state1;
    }
}
  
```

도면5

```

public class TimeTransition inherits Transistion
{
    protected override string OutputString ()
    {
        return "It is" + "GetTime()";
    }
}

public class WeatherDialogue inherits Subdialogue
{
    private State state;
    private Transition TomorrowTrans, weekTrans;

    void Initialize ()
    {
        state = new State ("Do you want the forecast for
            tomorrow or next week");
        tomorrowTrans = new Transation (state, Exit,
            "For tomorrow",
            "It will be sunny and warm.");
        weekTrans = new Transition (state, Exit,
            "For next week",
            "It will be hot, around 30 degrees.")
    }
}
  
```