(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-513425
(P2004-513425A)
(43)公表日　平成16年4月30日(2004.4.30)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/44	G06F	9/06	620A	5B076
	G06F	9/06	620C	

審査請求 有　予備審査請求 有　（全 38 頁）

(21)出願番号	特願2002-539952(P2002-539952)
(86)(22)出願日	平成13年10月19日(2001.10.19)
(85)翻訳文提出日	平成15年4月28日(2003.4.28)
(86)国際出願番号	PCT/US2001/050119
(87)国際公開番号	WO2002/037268
(87)国際公開日	平成14年5月10日(2002.5.10)
(31)優先権主張番号	09/702,224
(32)優先日	平成12年10月31日(2000.10.31)
(33)優先権主張国	米国(US)
(81)指定国	EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),CA,JP

(71)出願人　591067923
ユニシス　コーポレイション
ＵＮＩＳＹＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　ペンシルバニア州　１９４２４　ブルーベル，　ピイー．オー．ボックス　５００　タウンシップ　ライン　アンド　ユニオン　ミーテイング　ローズ（番地なし）
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄
(74)代理人　100083703
弁理士　仲村　義平

最終頁に続く

(54)【発明の名称】ダイアログフローインタープリタ開発ツール

(57)【要約】
コンピュータソフトウェア製品（２００）を用いて、人間とコンピュータとの間のダイアログを可能にするためのアプリケーションを作成する。ソフトウェア製品は、開発者が、ダイアログフローインタープリタ（２３２）への一般化命令を明記できるようにすることにより、時間がかかりかつ技術的に難しいプログラミングタスクからソフトウェア開発者を解放するプログラミングツール（２１０）を提供する。ダイアログフローインタープリタ（２３２）は、スピーチアプリケーションを実現する機能を呼出して、他のアプリケーションに利用可能なダイアログオブジェクトを自動的にライブラリ（５４０）に入れる。ＤＦＩによって作成されたスピーチアプリケーションは、ＣＯＭ（コンポーネントオブジェクトモデル）オブジェクトとして実現され得るため、アプリケーションをさまざまな異なるプラットフォームに容易に統合することができる。さらに、「トランスレータ」オブジェクトクラスを設けて、通貨、数値データ、日付、時間、列変数などの特定のタイプのデータを扱う。これらのトランスレータオブジェクトクラスは、ＤＦＩライブラリの一部、または、ダイアログ実現とは別個のサブライブラリのいずれかとしてのユーティリティを有する。

【特許請求の範囲】
【請求項１】
人間とコンピュータとが対話できるようにする、コンピュータ上で実行するための、ダイアログによって使用可能にされるアプリケーションを開発する方法であって、
（ａ）会話フローを特定する命令を設計ツールに入力するステップを含み、前記設計ツールはデータファイルを作成し、前記データファイルは、人間－コンピュータ間の、スピーチで使用可能にされる対話を実現するため、プロンプト、応答、ブランチおよび会話フローに関する情報を含み、さらに、
（ｂ）前記アプリケーション内でインタープリタオブジェクトを具現化するステップを含み、前記インタープリタオブジェクトは、データファイルを解釈して、前記データファイルが規定する、人間－コンピュータ間のダイアログによって使用可能にされる対話を提供する、方法。
【請求項２】
前記データファイルは、音声認識エンジンに関する情報をさらに含む、請求項１に記載の方法。
【請求項３】
前記データファイルは自動的に保存される、請求項１に記載の方法。
【請求項４】
前記命令を入力するステップはグラフィカルインターフェースを通して行われる、請求項１に記載の方法。
【請求項５】
人間とコンピュータとが対話できるようにする、コンピュータ上で実行するための、ダイアログで使用可能にされるソフトウェアを開発するためのシステムであって、
会話フローを特定する命令を受入れるための設計ツールを含み、前記設計ツールはデータファイルを作成し、さらに
前記データファイルを解釈するためのインタープリタを含み、前記インタープリタは人間とコンピュータとの対話を自動的に可能にする、システム。
【請求項６】
前記システムはライブラリをさらに含み、ライブラリは前記データファイルを含む、請求項５に記載のシステム。
【請求項７】
設計ツールはグラフィカルインターフェイスをさらに含む、請求項５に記載のシステム。
【請求項８】
コンピュータが実行可能な命令を含むコンピュータ読み取り可能記録媒体であって、命令はコンピュータに、
人間とコンピュータとの間の会話フローを特定する命令を受け入れるステップと、
インタープリタに入力するためのデータを生成するステップと、
前記データファイルを解釈するステップと、
人間とコンピュータとのダイアログによって使用可能にされる対話を提供するステップとを実行させる、コンピュータ読み取り可能な記録媒体。
【請求項９】
他のソフトウェア開発者が一般化コードに即座にアクセスできるようにする命令をさらに含む、請求項８に記載のコンピュータ読み取り可能な記録媒体。
【請求項１０】
人間とコンピュータとの間でダイアログを行うための、コンピュータで実現されるシステムで用いるためのダイアログフローインタープリタ（ＤＦＩ）であって、ＤＦＩは、人間－コンピュータ間のダイアログを実現するための、プロンプト、応答、ブランチおよび会話フローに関する情報を含むデータファイルを読出すためのコンピュータ実行可能な命令と、共有オブジェクトのライブラリと組合わせて前記情報を用いて前記ダイアログを行うためのコンピュータ実行可能なコードとを含む、ダイアログフローインタープリタ。

【請求項１１】
前記ＤＦＩは、ＤＦＩに加えて、言語インタープリタ、認識エンジンおよび音声入出力装置を含むアプリケーションで実現される、請求項１０に記載のＤＦＩ。
【発明の詳細な説明】
【０００１】
【発明の分野】
この発明は、一般的には、スピーチで使用可能にされる対話型音声応答（ＩＶＲ）システムおよび人間とコンピュータとの間のダイアログに関わる同様のシステムに関する。より特定的には、この発明は、特定のタイプのデータ（たとえば、通貨、日付、列変数など）を扱うためのトランスレータオブジェクトクラスはもちろん、ダイアログの低レベルディテールを実現するためのダイアログフローインタープリタ開発ツールを提供する。
【０００２】
【発明の背景】
コンピュータは、我々の日常生活の至るところに見られるようになった。今日、コンピュータは、単なる計算を超える役割を行っている。すなわち、スーパーマーケットのスキャナは、店舗の在庫を調べながら食料品の金額を計算し；コンピュータ化された電話交換センターは何百万もの通話を管理し；自動預払機（ＡＴＭ）は、ほぼどこからでも人々が銀行取引を行えるようにする、などである。大部分の人々にとって、何らかの方法でコンピュータと対話しない日が一日でもあるとは想像しにくい。
【０００３】
以前は、コンピュータのユーザは、キーボードもしくはマウスまたはより最近では電話機のプッシュトーン（デュアルトーンマルチフリーケンシ、すなわちＤＴＭＦと称する）により、コンピュータに従ってコンピュータと対話しなければならなかった。しかし、コンピュータとの対話はより容易でかつより使いやすくなる傾向にある。コンピュータと人間との間の対話をより容易にする１つの方法は、話し言葉によって人間とコンピュータとが対話できるようになることである。
【０００４】
人間とコンピュータとの間のダイアログを可能にするため、まずコンピュータは、話し言葉を検知し、それを、単純なテキストなどの何らかのコンピュータ読み取り可能なデータの形態に変換する音声認識能力を必要とする。次に、コンピュータはコンピュータ読み取り可能なデータを解析し、それらの言葉がどういう意味で用いられたのかを判断する何らかの方法を必要とする。高レベルの、スピーチで作動させられるアプリケーション、音声で作動させられるアプリケーション、または、自然言語理解アプリケーションは、典型的に、ユーザとアプリケーションをホストするコンピュータシステムとの間で段階的に話されるダイアログを行うことによって作動する。従来の方法を用いると、そのような高レベルアプリケーションの開発者は、可能な各ダイアログおよび各ダイアログの各段階を実現するソースコードを特定する。エラー強さのあるアプリケーションを実現するため、開発者は、ソフトウェアにおいて、可能な各プロンプトに対する可能な各ユーザ応答を、たとえそのような応答が予期されたものであってもなくても、予想し、かつ、扱う。そのような低レベルディテールを扱う高レベル開発者にかかる負担はかなり大きい。
【０００５】
スピーチで使用可能にされるアプリケーションに対する要求が高まるにつれ、開発リソースに対する要求も高まっている。現在、スピーチで使用可能にされるアプリケーションに対する要求は、アプリケーションをコード化するのに利用可能な開発リソースを超えている。また、アプリケーションを書くのに必要な専門知識を持つ開発者に対する要求も、その専門知識を持つ開発者の許容能力を超えている。したがって、対話型スピーチアプリケーションを開発するプロセスを単純化し、かつ、促進する必要性が存在する。
【０００６】
スピーチで使用可能にされるアプリケーションを開発し、かつ、これらのシステムを開発するのに必要なスキルレベルを上げるのにかかる時間の長さに加え、スピーチで使用可能

にされるアプリケーション開発の現在のモードのそれ以上の欠点は、ベンダーに固有であること、すなわち、ベンダーが変わるとコードの再利用が大きく阻害されてしまうことと、特定用途向けであること、すなわち、既に書かれたコードを別のアプリケーションに再利用することができないこととである。したがって、ベンダーから独立したシステムと、再利用可能なコードとを作成可能にする必要性もある。

【０００７】

ＩＶＲシステムに関するさらなる背景は、２０００年７月２５日の「スピーチで使用可能にされるアプリケーションのためのシステムおよび方法」（“Ｓｙｓｔｅｍ ａｎｄ Ｍｅｔｈｏｄ ｆｏｒ Ｓｐｅｅｃｈ Ｅｎａｂｌｅｄ Ａｐｐｌｉｃａｔｉｏｎ”）と題された米国特許第６，０９４，６３５号、１９９９年１１月３０日の「スプレッドシートまたはテーブルインターフェイスを用いて言語グラマーを作成するためのシステムおよび方法」（“Ｓｙｓｔｅｍ ａｎｄ Ｍｅｔｈｏｄ ｆｏｒ Ｃｒｅａｔｉｎｇ ａ Ｌａｎｇｕａｇｅ Ｇｒａｍｍａｒ ｕｓｉｎｇ ａ Ｓｐｒｅａｄｓｈｅｅｔ ｏｒ Ｔａｂｌｅ Ｉｎｔｅｒｆａｃｅ”）と題された米国特許第５，９９５，９１８号、ならびに、１９９９年１０月２９日に出願された、「タスク指向ダイアログモデルおよびマネージャ」（“Ｔａｓｋ Ｏｒｉｅｎｔｅｄ Ｄｉａｌｏｇ Ｍｏｄｅｌ，ａｎｄ Ｍａｎａｇｅｒ”）と題された米国特許出願番号第０９／４３０，３１５号に見出すことができる。これらの特許は、ユニシス・コーポレイション（Ｕｎｉｓｙｓ Ｃｏｒｐｏｒａｔｉｏｎ）に同一に譲渡される。

【０００８】

【発明の概要】

この発明は、人間とコンピュータとの間のダイアログを可能にするためのアプリケーションを作成するのに用いられるコンピュータソフトウェア製品に関する。ただし、この発明は必ずしもそれらに限定されるわけではない。そのようなアプリケーションは、（バンキング、証券業務またはインターネット上などでの使用を含み）どの業界で使用されてもよく、ユーザは、たとえば、電話、携帯電話またはマイクなどを用いてコンピュータとダイアログを行う。

【０００９】

この発明は、開発者が、ダイアログフローインタープリタ開発ツールまたはＤＦＩツールに一般化命令を明記できるようにすることにより、時間がかかりかつ技術的に難しい開発タスクからソフトウェア開発者を解放する開発ツールを提供することによって上記必要性を満たす。アプリケーションはオブジェクト（すなわちＤＦＩオブジェクト）を具現化し（ｉｎｓｔａｎｔｉａｔｅ）、次にオブジェクトは、スピーチアプリケーションを実現する機能を呼出す。ＤＦＩツールは、他のアプリケーションに利用可能なダイアログオブジェクトを自動的にライブラリに入れる。

【００１０】

ＤＦＩツールによって作成されたスピーチアプリケーションは、ＣＯＭ（コンポーネントオブジェクトモデル）オブジェクトとして実現してもよいため、アプリケーションをさまざまな異なるプラットフォームに容易に統合することができる。多数の異なる音声認識エンジンもサポートされ得る。特定のアプリケーションで用いられる特定の音声認識エンジンは、容易に変更可能である。

【００１１】

この発明の別の局面は、通貨、数値データ、日付、時間、列変数などの特定のタイプのデータを扱うように設計された「トランスレータ」オブジェクトクラスを提供することである。これらのトランスレータオブジェクトクラスは、ダイアログを実現するための上記オブジェクトのＤＦＩライブラリの一部またはダイアログ実現とは別個のサブライブラリのいずれかとしてのユーティリティを有し得る。

【００１２】

この発明の他の局面は以下に記載される。

【００１３】

【好ましい実施例の詳細な説明】
概要
図１は、従来のＩＶＲ型システムを示す。そのようなシステムでは、人（図示せず）はサーバコンピュータ１１０と通信する。サーバコンピュータ１１０はデータベース記憶システム１１２に結合され、データベース記憶システム１１２は、発信者とダイアログを行う際にサーバコンピュータ１１０の動作を制御するためのデータおよびコードを含む。示されるように、サーバコンピュータ１１０は、電話機１１６などの電話機を介して発信者へのアクセスを提供する公衆電話網（ＰＳＴＮ）１１４に結合される。述べられたように、スピーチによって使用可能にされるそのようなシステムは、ボイスメール、コールセンタ、バンキングなどを含む多様な用途で用いられる。
【００１４】
以前は、スピーチアプリケーション開発者は、音声認識エンジンを選択し、アプリケーション特有の音声認識エンジン特有システムをコード化する。この場合、開発者は、可能な事象の全ユニバースを予想して提供し、ダイアログのありとあらゆるディテールを扱う必要がある。そのようなアプリケーションは、新たなアプリケーションに対してはすべて書換えなければならないか、または、異なる音声認識エンジンを用いなければならない。
【００１５】
先行技術とは反対に、図２を参照して、この発明は、プログラマが使いやすいグラフィカルインターフェイス（図示せず）を通してアクセス可能なダイアログフローインタープリタ（ＤＦＩ）設計ツール２１０に対する（会話の多くのステート（ｓｔａｔｅ）またはターン（ｔｕｒｎ）を潜在的に含む）会話フローについての一般化命令を開発者が明記できるようにすることにより、時間のかかる低レベルプログラミングタスクから開発者を解放するシステムを提供する。ＤＦＩ設計ツール２１０はデータファイル２２０（アプリケーションのシェル）を作成する。開発者がさまざまなプログラミング言語で書くことのできる呼出プログラム（スピーチアプリケーション）２３０が実行されると、呼出プログラム２３０はダイアログフローインタープリタ２３２を具現化して、ＤＦＩ設計ツール２１０が作成したデータファイル２２０をインタープリタ２３２に与える。次にダイアログフローインタープリタ２３２は、ＤＦＩオブジェクトの機能を呼出してスピーチアプリケーションを実現し、以前はプログラマが書かなければならなかったステートハンドリングおよび会話フローのあらゆるディテールを与える。呼出プログラム２３０は、一旦書いてしまえば、異なるアプリケーションに使用可能である。アプリケーションは、プロンプトおよび予期される応答の内容において、ならびに、結果的に生じる処理（ブランチおよび会話フロー）において、ならびに、用いられる音声認識エンジンにおいて、互いに異なる。この発明に従うと、これらはすべてデータファイル２２０に保存され得る。したがって、データファイル２２０を変更することにより、既存の呼出プログラム２３０を異なるアプリケーションに使用可能である。
【００１６】
開発ツール２００は、ダイアログオブジェクトを含むどのレベルのディテールの再利用可能コードも、他のアプリケーションでの使用のためにアクセス可能にされ得るライブラリに自動的に保存する。ダイアログオブジェクトは、ステートを共にリンクさせるのに関わる処理を含む１つ以上のダイアログステートの集合である。
【００１７】
開発プログラミングツールによって作成されたスピーチアプリケーションは実行可能オブジェクトとして実現されるため、アプリケーションはさまざまな異なるプラットフォームに容易に統合可能である。多数の異なる音声認識エンジンがサポートされ得る。特定のアプリケーションで用いられる特定の音声認識エンジンは容易に変更可能である。この発明と先行技術とを比較することにより、この発明をより詳細に説明する。
【００１８】
先行技術
再び図１を参照して、ダイアログベースのシステムにおいてユーザがコンピュータと通信

する最も一般的な方法は、マイクを通してであるか、または、電話交換システム１１４により、データベース１１２に人間とコンピュータが対話できるようにするソフトウェアを保存するコンピュータに接続された電話機１１６を通してである。コンピュータがユーザから特定の情報を引き出そうとする、コンピュータとユーザ間の各々の対話がステートまたはターンと呼ばれる。各ステートにおいて、コンピュータはプロンプトでスタートし、ユーザは口頭で応答する。アプリケーションは、ユーザが言ったことを認識および解釈し、その応答に基づいて適切な行動を取り、次に、会話を次のステートまたはターンに進めなければならない。このステップは以下のとおりである。

【００１９】

１．コンピュータがプロンプトを発する。

２．ユーザ（または発信者）が応答する。

【００２０】

３．音声レコグナイザが、応答をコンピュータが読出可能な形態に変換する。

４．アプリケーションが応答を解釈し、これに応じて作動する。これは、たとえば、問合せに対するデータベースアクセスを含み得る。

【００２１】

５．アプリケーションがユーザに応答し得る。

６．ユーザから満足な応答を受取るまで、ステップ１から５を繰返し得る。

【００２２】

７．アプリケーションが次のステートに遷移する。

したがって、ダイアログベースのスピーチアプリケーションは、ユーザをゴールに導く一組のステートを含む。以前は、開発者は、ダイアログにおける各ステップをコード化して、可能な事象のユニバースにおいて、可能な各事象および可能な各応答ごとに、時間がかかりかつ技術的に複雑なタスクをコード化しなければならなかった。開発者は、たとえばＰａｒｉｔｙなどの対話型音声応答（ＩＶＲ）システムを選択して、Ｎｕａｎｃｅ、ＬｅｒｎｏｕｔおよびＨａｕｓｐｉｅなどの音声認識エンジンまたはＩＶＲ環境にプラグインする別の音声認識エンジンを用いて、その言語と関連のプログラミング言語でアプリケーションをコード化しなければならなかった。

【００２３】

音声オブジェクトは市販されている。図３を参照して、音声オブジェクト３２２，３２４は、典型的には、プロンプト、グラマーおよび応答である言語行為に入るあらゆるものが予めパッケージされたビットである。この機構において、たとえば、Ｇｅｔ Ｓｏｃｉａｌ Ｓｅｃｕｒｉｔｙ Ｎｕｍｂｅｒ３２２などの音声オブジェクトをベンダーから購入する。開発者は、選択された音声オブジェクトに必要なプログラミング言語でソフトウェアコード３２０を書き、購入したＧｅｔ Ｓｏｃｉａｌ Ｓｅｃｕｒｉｔｙ Ｎｕｍｂｅｒ音声オブジェクト３２２を自分のソフトウェアに入れる。プログラムが実行され、社会保険番号が必要な段階に達すると、Ｇｅｔ Ｓｏｃｉａｌ Ｓｅｃｕｒｉｔｙ Ｎｕｍｂｅｒ音声オブジェクト３２２が呼出される。アプリケーションは、どのように質問が行われるかを僅かに変更したかもしれないが、音声オブジェクトの柔軟性の範囲は限られている。ユーザからの応答を得た後、コントロールはアプリケーション３２０に戻される。次に、開発者が書いたアプリケーション３２０は、次のステート、すなわちＧｅｔ ＰＩＮ Ｎｕｍｂｅｒ３２４などへの遷移を扱わなければならない。音声オブジェクトは、特定の展開用（ｄｅｐｌｏｙｍｅｎｔ）システム（たとえば、Ｓｐｅｅｃｈ Ｃｈａｎｎｅｌｓと称されるＮｕａｎｃｅの「ＩＶＲシステム」および、アプリケーションフレームワークと称されるＳｐｅｅｃｈＷｏｒｋｓの「ＩＶＲシステム」）に対して実現される。これら再利用可能なものは、それらが構築された環境内においてのみ再利用可能である。たとえば、Ｄｉａｌｏｇ Ｍｏｄｕｌｅｓと呼ばれるＳｐｅｅｃｈＷｏｒｋｓのこの実現例は、ＳｐｅｅｃｈＷｏｒｋｓのアプリケーションフレームワーク内でのみ機能する。コアロジックは、実行プラットフォームに結合されているため、再利用不可能である。

【００２４】

ＤＦＩ設計ツール
これに対し、この発明に従うと、図４を参照して、開発者はＤＦＩ設計ツール４００を用いて、Ｇｅｔ Ｓｏｃｉａｌ Ｓｅｃｕｒｉｔｙ Ｎｕｍｂｅｒ、Ｇｅｔ ＰＩＮ Ｎｕｍｂｅｒなどの多くのそのようなステートを含むステップ４１０に示されるようなアプリケーション全体の設計を入力する。アプリケーションが一旦シミュレータでリハーサルされると（参考特許連続番号第０９／４１７，１６６号を参照）（ステップ４２０）、その設計を表わすファイルが生成され得る（ステップ４４０および４５０）。
【００２５】
図５に示されるように、さまざまなプログラミング言語のうちいずれかで開発者がコード化したソフトウェアアプリケーション５１０は、ダイアログフローインタープリタ５３０を具現化し、これに対して、ＤＦＩ設計ツールが上記で生成したファイル５２０中で特定された設計を解釈するように命令する。ダイアログフローインタープリタ５３０は、アプリケーションを通してフローを制御して、以前は開発者が書かなければならなかった基本コード５４０をすべて供給する。
【００２６】
図６Ａの６１２および図６Ｂの６２２からわかるように、プログラマが書かなければならないコードの量は実質的に低減される。実際に、いくつかのアプリケーションでは、これがまったくなくなってしまう可能性がある。
【００２７】
ダイアログフローインタープリタ
この発明のダイアログフローインタープリタまたはＤＦＩは、ダイアログの低レベルディテールを実現する「標準化された」オブジェクトのライブラリを提供する。ＤＦＩは、スピーチアプリケーションの実現を単純化するアプリケーションプログラミングインターフェイス（ＡＰＩ）として実現され得る。スピーチアプリケーションは、ＤＦＩ開発ツールと称されるツールを用いて設計され得る。この発明がもたらす単純化は、ＤＦＩが始めから終わりまで自動的にスピーチアプリケーションの全ダイアログを駆動可能であることによるものであり、これにより、ダイアログ管理のきわめて重要でかつしばしば複雑なタスクが排除される。伝統的に、そのようなプロセスはアプリケーションに依存するので、新しいアプリケーションごとに再実現が必要である。ＤＦＩは、１回の書込で何度も実行することができる（ｗｒｉｔｅ－ｏｎｃｅ，ｒｕｎ－ｍａｎｙ）方策を提供することにより、この問題を解決する。
【００２８】
図２は、ＤＦＩ設計ツール２１０、ダイアログフローインタープリタ２３２、および、他のスピーチアプリケーションコンポーネントの間の関係を図示する。（この図では、ブロック状の矢印はデータフローの方向を示す。）
機能的要素
スピーチアプリケーションは、ステート間の一連の遷移を含む。各ステートは、再生すべきプロンプト、（音声システムのユーザが何を発言し得るかを聞くために）ロードすべき音声レコグナイザのグラマー、発信者の応答に対する返事、および、各応答に基づいて取るべき行動を含む、それ自身のプロパティの組を有する。ＤＦＩは、アプリケーションの使用期間を通じていずれの所与の時点でもダイアログのステートを記録し、ステートプロパティにアクセスする機能を呈する。
【００２９】
図７を参照して、ステートプロパティは、「共有オブジェクト」７１０と称されるオブジェクトに保存されることがわかる。これらのオブジェクトの例には、Ｐｒｏｍｐｔオブジェクト、Ｓｎｉｐｐｅｔオブジェクト、Ｇｒａｍｍａｒオブジェクト、Ｒｅｓｐｏｎｓｅオブジェクト、Ａｃｔｉｏｎオブジェクトおよびｖａｒｉａｂｌｅオブジェクトが含まれるが、これらに限定されるものではない。
【００３０】
例示的なＤＦＩ機能７８０は、上記のオブジェクトのうちいくつかを返す。これらの機能

は以下のものを含む。
【００３１】
ＧＥＴ－ＰＲＯＭＰＴ７２０：再生すべき適切なプロンプトを返す。次に、このプロンプトは、サウンド出力のため、適切なサウンド再生ルーチンに送られる。
【００３２】
ＧＥＴ ＧＲＡＭＭＡＲ７３０：現在のステートに対する適切なグラマーを返す。次にこのグラマーは音声認識エンジンにロードされる。
【００３３】
ＧＥＴ ＲＥＳＰＯＮＳＥ７４０：実際のユーザ応答、この応答が含み得る何らかの変数、および、この応答に対して規定されるすべての可能な行動を含む応答オブジェクトを返す。
【００３４】
ＡＤＶＡＮＣＥ－ＳＴＡＴＥ７５０：ダイアログを次のステートに遷移させる。
他のＤＦＩ機能を用いてステート独立プロパティ（すなわち、グローバルプロジェクトプロパティ）を検索する。これらは以下のものを含むが、これらに限定されるものではない。
【００３５】
プロジェクトの経路７６０
プロジェクトのサウンド経路
入力モード（ＤＴＭＦまたは音声）
バージインモード（ＤＴＭＦまたは音声）
現在のステート
以前のステート。
【００３６】
ＤＦＩの代替的使用
ダイアログ計測のためのロギング装置 － ＤＦＩはステート間の遷移の内部（ｉｎｔｅｒｎａｌｓ）を制御するので、たとえば、あるステートに何回入ったかを数えて、どのようにスピーチアプリケーションが用いられるか、または、どのようにスピーチアプリケーションが動作するかに関する統計を収集し得るのは容易なことである。
【００３７】
スピーチアプリケーションストレステスタ － ＤＦＩはステート間の遷移の内部を制御するので、ＤＦＩツールは、ダイアログのコンピュータ側だけでなくダイアログの人間側を設けることによりスピーチアプリケーションのテストを容易にする、（テキスト－音声変換を用いる）アプリケーションの開発を可能にする。
【００３８】
図７は、アプリケーションプログラミングインターフェイス（ＡＰＩ）としてＤＦＩ機能７８０がどのように実現され得るか、または、どのように見えるかを図示する。
【００３９】
ＤＦＩと音声オブジェクトとの比較
音声オブジェクト（業界では一般的概念）は、典型的に、プロンプト（言うべきこと）、グラマー（聞くべきこと）、および、おそらくは、システム側の何らかの反応など、「言語行為」に入るあらゆるものが予めパッケージされたビットである。これは、（うまくいかない可能性があるあらゆることを考えるまでは単純に思われる）単一ビットの情報の集まりを含み得る。１つの方策は、予めパッケージされた機能（たとえば、ＳｐｅｅｃｈＷｏｒｋｓ（ｗｗｗ．ｓｐｅｅｃｈｗｏｒｋｓ．ｃｏｍ））を提供することである。基本モデルの例は以下のとおりである。すなわち、設計者は、Ｇｅｔ Ｓｏｃｉａｌ Ｓｅｃｕｒｉｔｙ Ｎｕｍｂｅｒと呼ばれる音声オブジェクトを（たとえばＮｕａｎｃｅから）購入し、これを自分のプログラムに入れる。プログラムがユーザの社会保険番号が必要な段階に達すると、設計者はＧｅｔ Ｓｏｃｉａｌ Ｓｅｃｕｒｉｔｙ Ｎｕｍｂｅｒオブジェクトを呼出す。アプリケーションは、正確には質問がされる方法を変更することにより、ま

たは、それが聞こえる範囲を広げることにより、それを少し変えたかもしれないが、基本的な価値としては、オブジェクトの予めパッケージされた方法であり、かつ、予め調整された機能である。

【００４０】

この発明のダイアログフローインタープリタ開発ツールにおいて、設計者は、設計ツール（たとえば、ユニシス・コーポレイションが提供するＤＦＩツール）を用いて、（ＳＳ＃入手およびＰＩＮ入手などの多くのステートを潜在的に含む）アプリケーション全体の設計を入力する。このアプリケーションがシミュレータ（ウィザード・オブ・オズ・テスタ（Ｗｉｚａｒｄ ｏｆ Ｏｚ ｔｅｓｔｅｒ））で一旦リハーサルされると、その設計（たとえばＭｙＳｐｅｅｃｈＡｐｐ）を表わすファイルが生成される。ＤＦＩは、（何らかのプログラミング言語で書かれた）「ランタイム」アプリケーションによって具現化され、設計ツールが作成した設計（ＭｙＳｐｅｅｃｈＡｐｐ）を解釈するように命令される。一旦セットアップされると、アプリケーションコードは、何が起こっているかの詳細をＤＦＩに与え、次に何をすべきかについて設計を「逆読み（ｒｅａｄ ｂａｃｋ）」しさえすればよい。したがって、たとえば、設計者は以下のものなどのシーケンスを指示し得る。

【００４１】

あなたのＳＳ番号は何番ですか？
（ＳＳ番号を聞く）
あなたのＰＩＮは何ですか
（ＰＩＮを聞く）
オーダを希望しますか、または、問題の報告を希望しますか
（ＯＲＤＥＲまたはＲＥＰＯＲＴ＿Ａ＿ＰＲＯＢＬＥＭを聞く）
ＯＲＤＥＲである場合、
オーダは何ですか
そうではなく、ＲＥＰＯＲＴ Ａ ＰＲＯＢＬＥＭである場合、
問題は何ですか。

【００４２】

この場合、ＤＦＩはまず、どのプロンプトを再生すべきかをプログラムが尋ねると、これは「あなたのＳＳ番号は何番ですか？」と返すステートに入り、プログラムがＳＳ＃を聞く必要があることを示す。これが達成され、次に進むようにアプリケーションが一旦ＤＦＩに命令すると、アプリケーションは、ＤＦＩに何と言うべきかを再び尋ね、今度は「あなたのＰＩＮは何ですか」と返す。ＤＦＩはアプリケーションが終了するまで方向性のあるデータを供給し続ける。重要なのは、ＤＦＩが、ダイアログ（プロンプト、何を聞くべきか、など）の各ターンごとに「内部」を供給し、アプリケーションを通してフローを供給することである。

【００４３】

同様の問題を扱うものの、ＤＦＩは音声オブジェクトモデルとは大きく異なっている。音声オブジェクトは、プログラムが無効にし得るデフォルトをセットアップする（プログラムはどこかからこのことを知っていなければならない）一方、ＤＦＩは、次に何をすべきかをアプリケーションに与える。音声オブジェクトは柔軟性がなくかつ予めプログラムされ、範囲が限られているが、ＤＦＩはアプリケーション全体に対して構築され、かつダイナミックである。音声オブジェクトは特殊な目的のために「調整」される。上述のこの調整は、ＤＦＩ設計ツールを通して提供され得る。相違点を別のやり方で考えると、ＤＦＩは、どのように能力を共に「リンクする」かを含め、ツールによって構築される「カスタム」スピーチ能力を与える。音声オブジェクトは、（「専門家の設計」および調整という利点を有する）「予めパッケージされた」能力を与えるが、それらの間に「フロー」は存在しない。

【００４４】

トランスレータオブジェクトクラス
スピーチアプリケーションは、ソフトウェアが解釈可能な形態で情報を検索できなければ

ならない。一旦情報を入手すると、その情報を特定のスピーチフォーマットで外部に出力することが望ましいであろう。この発明に従うと、トランスレータオブジェクトクラスは、特定の情報をどのように出力すべきかについて詳細を特定するパラメータを開発者が提供できるようにし、ＤＦＩはそのタスクを実行するのに必要なあらゆるものを返す。たとえば、所望のオブジェクトが、標準時での現在のベルギーの時刻を英語で出力することである場合、開発者は、言語（英語）、地域（ベルギー）、時刻（ベルギーの現在時刻）およびフォーマット（標準時）を特定し、ＤＦＩは、それらの特徴（英語で話される、標準的フォーマットでの現在のベルギーの時刻）を有するデータ構造をリスナが聞けるようにするのに必要なあらゆるもののプレイリストを返す。

【００４５】

たとえば、ＤＦＩがプロンプティングを完了しているとき、ＤＦＩは図７の７２０のＧＥＴ ＰＲＯＭＰＴ機能にアクセスし、（出力されたスピーチが記録済ファイルである場合は）以下のものを返す。

【００４６】

１．”Ｉｔ ｉｓ ｎｏｗ”．ｗａｖ ファイル
２．時間の例の値（変数）である１２：３５ｐｍおよび関連ファイルである
ｔｗｅｌｖｅ．ｗａｖ
ｔｈｉｒｔｙ．ｗａｖ
ｆｉｖｅ．ｗａｖ
ｐｍ．ｗａｖ
３．”ｉｎ Ｂｅｌｇｉｕｍ” ｗａｖ ファイル。

【００４７】

リスナには、”Ｉｔ ｉｓ ｎｏｗ ｔｗｅｌｖｅ ｔｈｉｒｔｙ－ｆｉｖｅ ｐｍ ｉｎ Ｂｅｌｇｉｕｍ．”と聞こえるであろう。なお、上記例は例示の目的のためのみであることを理解されたい。この発明は、テキスト－音声変換（コンピュータ生成）音声出力も含む。

【００４８】

これに代えて、開発者が、ＤＦＩを用いずに自分のアプリケーションで直接にオブジェクトを用いることを希望する場合、アプリケーションは、トランスレータに直接にアクセスし得る。トランスレータは、時間の例（１２：３５）の値および以下の関連ファイルを返す。

【００４９】

ｔｗｅｌｖｅ．ｗａｖ
ｔｈｉｒｔｙ．ｗａｖ
ｆｉｖｅ．ｗａｖ
ｐｍ．ｗａｖ。したがって、トランスレータオブジェクトクラスは、開発者が書いたスピーチアプリケーションまたはＤＦＩが使用可能なオブジェクトを含む。

【００５０】

市販の音声オブジェクトは同様の機能を提供し得るが、トランスレータオブジェクトクラスの進歩性は、クラスに追加すべき開発者自身のオブジェクトを自分で書くことができるため、開発者が、情報の出力のされ方の低レベルディテールの制御を失わないことにある。開発者が市販の音声オブジェクトを用いる場合、開発者は、音声オブジェクトが機能する態様を制御する柔軟性が失われるのを受入れなければならない。この発明に従うトランスレータオブジェクトの場合、開発者は、最大限の自動化を依然として入手しながら、低レベルディテールの制御を維持する。

【００５１】

結論

要約すると、この発明は、ＩＶＲシステムなどにおいて、人間とコンピュータとの間の対話ダイアログを発生させるシステムおよび方法を提供する。しかしながら、この発明は、さまざまな変形および代替的構造が可能であることを理解されたい。この発明を本明細書

中に記載の特定の構造に限定する意図はない。これに対し、この発明は、この発明の範囲および意図に入るあらゆる変形、代替的構造および均等物を含むことが意図される。たとえば、この発明は、コンピュータと人間とが対話する、スピーチで使用可能にされないアプリケーションをサポートしてもよい。この発明は、テキストで表示され得るプロンプトのテキスト記述を再現して、ユーザが編集ボックスにタイプすることによって応答できるようにする。言い換えると、この発明の核になるのは、ダイアログの実現ではなく、各ステートのプロパティおよびダイアログフローである。そのような実施例は、コンピュータゲームにおいて、またはコンフィギュレーション情報を収集するソフトウェア内で、または単純なグラフィカルユーザインターフェイス（ＧＵＩ）技術が可能にするよりも対話的なインターネットアプリケーションにおいて利用され得る。

【００５２】

また、留意すべきなのは、この発明は、さまざまなコンピュータ環境で実現され得ることである。たとえば、この発明は、Ｊａｖａ（Ｒ）において実現されて、どのＪａｖａ（Ｒ）プログラミング言語からの直接アクセスも可能にし得る。さらに、実現例はＣＯＭ層で重ねられ得る（ｗｒａｐｐｅｄ）ので、ＣＯＭをサポートするどの言語もこの機能にアクセスすることができ、これにより、Ｖｉｓｕａｌ Ｂａｓｉｃ（Ｒ）、Ｃ／Ｃ＋＋などの伝統的な開発環境がこの発明を利用できるようにになる。この発明は、Ｗｏｒｄ、Ｅｘｃｅｌなどを含むがそれらに限定されないＭｉｃｒｏｓｏｆｔ（Ｒ）内のアプリケーションから、たとえば、ビジュアルベーシック・フォー・アプリケーション（Ｖｉｓｕａｌ Ｂａｓｉｃ ｆｏｒ Ａｐｐｌｉｃａｔｉｏｎｓ（ＶＢＡ））までもアクセス可能であり得る。市販の、たとえばＰａｒｉｔｙなどの伝統的なＤＴＭＦ指向システムは、この発明をそれらのプラットフォームに埋込んでもよい。この発明およびその関連のオブジェクトは、ワールドワイドウェブおよびインターネット用の開発環境で展開されてもよく、これにより、ハイパーテキストマークアップ言語（ＨＴＭＬ）および同様のプロトコルがＤＦＩ開発ツールおよびそのオブジェクトにアクセスできるようになる。

【００５３】

本明細書中に記載のさまざまな技術は、ハードウェア、ソフトウェアまたはその両者の組合わせで実現され得る。好ましくは、この技術は、プロセッサ、（揮発性および不揮発性メモリおよび／または記憶素子を含む）プロセッサが読出し可能な記憶媒体、少なくとも１つの入力装置ならびに少なくとも１つの出力装置を各々が含むプログラマブルコンピュータ上で実行されるコンピュータプログラムにおいて実現される。プログラムコードは、入力装置を用いて入力されるデータに適用されて上述の機能を果たし、出力情報を生成する。出力情報は、１つ以上の出力装置に適用される。各プログラムは、好ましくは、高レベル手続き型プログラミング言語またはオブジェクト指向プログラミング言語で実現されてコンピュータシステムと通信する。しかしながら、プログラムは、所望により、アセンブリ言語または機械語で実現可能である。いずれにせよ、言語は、コンパイルされたまたは解釈された言語であり得る。各々のそのようなコンピュータプログラムは、好ましくは、コンピュータが記憶媒体または記憶装置を読出して上述の手順を行う際にコンピュータを設定しかつ作動させるための汎用または特殊目的プログラマブルコンピュータによって読出可能な記憶媒体または記憶装置（たとえば、ＲＯＭまたは磁気ディスク）に保存される。このシステムは、コンピュータプログラムを用いて構成されるコンピュータ読み取り可能な記憶媒体として実現されると考えられてもよい。ここでは、そのように構成される記憶媒体は、予め規定された特定の態様でコンピュータを作動させる。

【００５４】

この発明の例示的な実現例が以上で説明されたが、当業者は、この発明の新規な教示および利点から実質的に逸脱することなく、例示的な実施例において多くのさらなる変形が可能であることを容易に認めるであろう。したがって、これらおよびすべてのそのような変形は、この発明の範囲内に含まれることが意図される。

【図面の簡単な説明】

【図１】従来のＩＶＲシステムを概略的に示す図である。

【図２】この発明に従う、スピーチアプリケーションの開発のための方法のフローチャートの図である。
【図３】先行技術のスピーチアプリケーションを示すフローチャートの図である。
【図４】この発明に従う、設計開発およびスピーチアプリケーション用データファイルの生成のための方法のフローチャートの図である。
【図５】この発明に従う、スピーチアプリケーションの生成のための方法のフローチャートの図である。
【図６Ａ】先行技術のシステムを用いて開発者が書いたコードの量と、この発明に従うシステムを用いて開発者が書いたコードの量との比較を示す図である。
【図６Ｂ】先行技術のシステムを用いて開発者が書いたコードの量と、この発明に従うシステムを用いて開発者が書いたコードの量との比較を示す図である。
【図７】この発明に従う、機能および共有オブジェクトを表わす概略図である。

【図１】

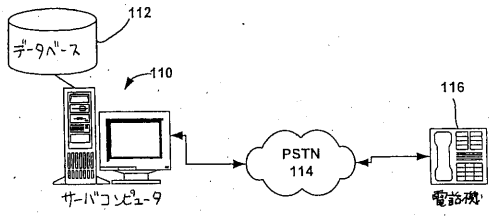

Figure 1 (Prior Art)

【図２】

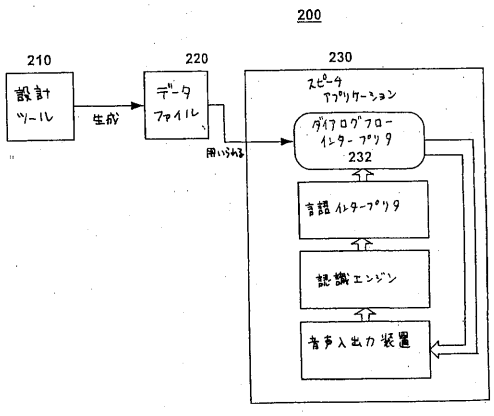

Figure 2
DFI 開発ツール

【図３】

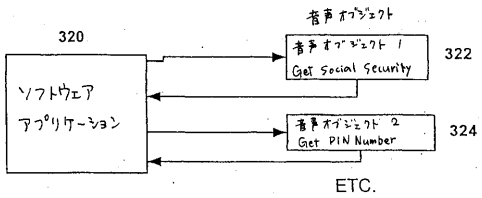

Figure 3 - Prior Art

【図４】

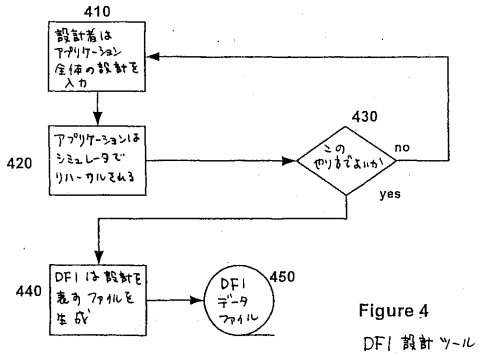

Figure 4
DFI 設計ツール

【図５】

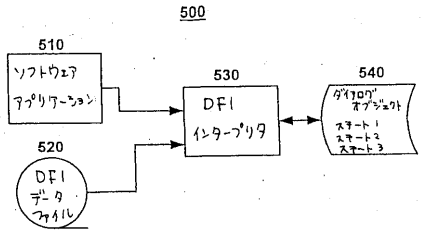

Figure 5
DFI

【図６Ａ】

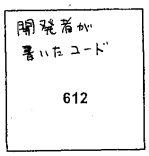

Figure 6(A) - Prior Art

【図６Ｂ】

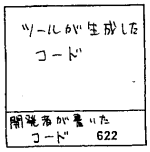

Figure 6(B)
DFI ツール

【図７】

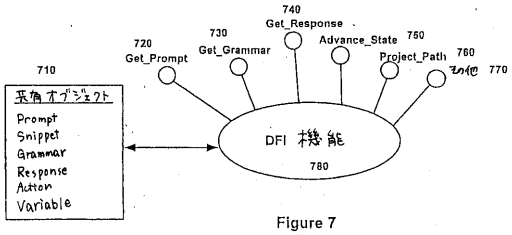

Figure 7
DFI 機能

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 May 2002 (10.05.2002)

PCT

(10) International Publication Number
WO 02/37268 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/US01/50119

(22) International Filing Date: 19 October 2001 (19.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/702,224 31 October 2000 (31.10.2000) US

(71) Applicant: **UNISYS CORPORATION** [US/US]; Township Line and Union Meeting Roads, P.O. Box 500, Blue Bell, PA 19424-0001 (US).

(72) Inventors: **SCHOLZ, Karl, Wilmer**; 203 Vassar Circle, Strafford, PA 19807 (US). **IRWIN, James, S.**; 55 Ream Road, Stevens, PA 17578 (US). **TAMRI, Samir**; 333 Lancaster Avenue, #309, Frazer, PA 19355 (US).

(74) Agents: **STARR, Mark, T.** et al.; Unisys Corporation, Township Line and Union Meeting Roads, P.O. Box 500, Blue Bell, PA 19424-0001 (US).

(81) Designated States *(national)*: CA, JP.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/37268 A2

(54) Title: DIALOGUE FLOW INTERPRETER DEVELOPMENT TOOL

(57) Abstract: A computer software product (200) is used to create applications for enabling a dialogue between a human and a computer. The software product provides a programming tool (210) that insulates software developers from time-consuming, technically-challenging programming tasks by enabling the developer to specify generalized instructions to a Dialogue Flow Interpreter (232), which invokes functions to implement a speech application, automatically populating a library (540) with dialogue objects that are available to other applications. The speech applications created through the DFI may be implemented as COM (component object model) objects, and so the applications can be easily integrated into a variety of different platforms. In addition, "translator" object classes are provided to handle specific types of data, such as currency, numeric data, dates, times, string variables, etc. These translator object classes have utility either as part of the DFI library or as a sub-library separate from dialogue implementation.

DIALOGUE FLOW INTERPRETER DEVELOPMENT TOOL

FIELD OF THE INVENTION

The present invention relates generally to speech-enabled interactive voice response (IVR) systems and similar systems involving a dialogue between a human and a computer. More particularly, the present invention provides a Dialogue Flow Interpreter Development Tool for implementing low-level details of dialogues, as well as translator object classes for handling specific types of data (e.g., currency, dates, string variables, etc.).

BACKGROUND OF THE INVENTION

Computers have become ubiquitous in our daily lives. Today, computers do much more than simply compute: supermarket scanners calculate our grocery bill while tracking store inventory; computerized telephone switching centers direct millions of calls; automatic teller machines (ATMs) allow people to conduct banking transactions from almost anywhere – the list goes on and on. For most people, it is hard to imagine a single day in which they will not interact with a computer in some way.

Formerly, computer users were forced to interact with computers on the computer's terms – by keyboard or mouse or more recently, by touch-tones on a telephone (called DTMF – for dual tone multi-frequency). More and more, however, the trend is to make interactions between computers easier and more user-friendly. One way to make interactions between computers and humans friendlier is to allow humans and computers to interact by spoken words.

To enable a dialogue between human and computer, the computer first needs a speech recognition capability to detect the spoken words and convert them into some form of computer readable data, such as simple text. Next the computer needs some way to analyze the computer-readable data and determine what those words, as they were used, meant. A high-level speech-activated, voice-activated, or natural language understanding application typically operates by conducting a step-by-step spoken dialogue between the user and the computer system hosting the application. Using conventional methods, the developer of such high-level applications specifies the source code implementing each possible dialogue, and each step of each dialogue. To implement a robust application, the developer anticipates and handles in software each possible user response to each possible prompt, whether such responses are expected or unexpected. The burden on the high-level developer to handle such low-level details is considerable.

As the demand for speech-enabled applications has increased, so has the demand on development resources. Presently, the demand for speech-enabled applications exceeds the development resources available to code the applications. Also, the demand for developers with the necessary expertise to write the applications exceeds the capacity of developers with that expertise. Hence, a need exists to simplify and expedite the process of developing interactive speech applications.

In addition to the length of time it takes to develop speech-enabled applications and the level of skill required to develop these systems, a further disadvantage of the present mode of speech-enabled application development is that it is vendor specific, significantly inhibiting reuse of the code if the vendor changes, and application specific, meaning that already written code can not be re-used for another application. Thus a need also exists to be able to create a system that is vendor-independent and code that is re-useable.

Additional background on IVR systems can be found in U.S. Patent No. 6,094,635, July 25, 2000, titled "System and Method for Speech Enabled Application"; in U.S. Patent No. 5,995,918, November 30, 1999, "System and Method for Creating a Language Grammar using a Spreadsheet or Table Interface" and in U.S. Patent Application No. 09/430,315 filed October 29, 1999, "Task Oriented Dialog Model, and Manager." These patents are commonly assigned to Unisys Corporation.

SUMMARY OF THE INVENTION

The present invention relates to but is not necessarily limited to computer software products used to create applications for enabling a dialogue between a human and a computer. Such an application might be used in any industry (including use in banking, brokerage, or on the Internet, etc.) whereby a user conducts a dialogue with a computer, using, for example, a telephone, cell phone or microphone.

The present invention satisfies the aforementioned needs by providing a development tool that insulates software developers from time-consuming, technically-challenging development tasks by enabling the developer to specify generalized instructions to the Dialogue Flow Interpreter Development Tool, or DFI Tool. An application instantiates an object (i.e. the DFI object), the object then invoking functions to implement the speech application. The DFI Tool automatically populates a library with dialogue objects that are available to other applications.

The speech applications created through the DFI Tool may be implemented as COM (component object model) objects, and so the applications can be easily integrated into a variety of different platforms. A number of different speech recognition engines may also be supported. The particular speech recognition engine used in a particular application can be easily changed.

Another aspect of the present invention is the provision of "translator" object classes designed to handle specific types of data, such as currency, numeric data, dates, times, string variables, etc. These translator object classes may have utility either as part of the DFI library of objects described above for implementing dialogues or as a sub-library separate from dialogue implementation.

Other aspects of the present invention are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 schematically depicts a conventional IVR system.

Figure 2 is a flowchart of a method according to the present invention for development of a speech application.

Figure 3 is a flowchart depicting a prior art speech application.

Figure 4 is a flowchart of a method according to the present invention for development of a design and the generation of a data file for a speech application.

Figure 5 is a flowchart of a method according to the present invention for generation of a speech application.

Figures 6(a) and 6(b) provide a comparison of the amount of code written by a developer using a prior art system to that written by a developer using a system in accordance with the present invention.

Figure 7 is a schematic diagram representing functions and shared objects in accordance with the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Overview

Figure 1 depicts a conventional IVR-type of system. In such a system, a person (not shown) communicates with a server computer, 110. The server computer, 110, is coupled to a database storage system, 112, which contains code and data for controlling the operation of the server computer, 110, in conducting a dialogue with the caller. As shown, the server computer, 110 is coupled to a public switched telephone network (PSTN), 114, which in turn provides access to callers via telephones, such as telephone, 116. As mentioned, such speech-enabled systems are used in a wide variety of applications, including voice mail, call centers, banking, etc.

Previously, speech application developers would choose a speech recognition engine and code an application-specific, speech recognition engine-specific system requiring the developer to handle each and every detail of the dialogue, anticipating and providing for the entire universe of possible events. Such applications would have to be completely rewritten for a new application or to use a different speech-recognition engine.

In contrast to the prior art, and referring to Fig. 2, the present invention provides a system that insulates developers from time-consuming, low-level programming tasks by enabling the developer to specify generalized instructions about the flow of a

conversation (potentially including many states or turns of a conversation), to a dialogue flow interpreter (DFI) design tool, 210, accessible through a programmer-friendly graphical interface (not shown). The DFI design tool, 210, produces a data file, 220, (a shell of the application). When the calling program (speech application), 230, which can be written by the developer in a variety of programming languages, executes, the calling program, 230, instantiates the dialogue flow interpreter, 232, providing to the interpreter, 232, the data file, 220, produced by the DFI design tool, 210. The dialogue flow interpreter, 232, then invokes functions of the DFI object to implement the speech application, providing all the details of state-handling and conversation flow that previously the programmer had to write. The calling program, 230, once written, can be used for different applications. Applications differ from one another in the content of prompts and expected responses and in resultant processing, (branches and conversation flow), and in the speech recognition engine used, all of which, according to the present invention, may be stored in the data file, 220. Therefore, by changing the data file, 220, the existing calling program, 230, can be used for different applications.

The development tool, 200, automatically saves reusable code of any level of detail, including dialogue objects, in a library that can be made accessible for use in other applications. A dialogue object is a collection of one or more dialogue states including the processing involved in linking the states together.

Because the speech applications created through the development programming tool are implemented as executable objects, the applications can be easily integrated into a variety of different platforms. A number of different speech recognition engines may be supported. The particular speech recognition engine used in a particular application can be easily changed. We will now explain the present invention in greater detail by way of comparing it with the prior art.

Prior Art

Referring again to Figure 1, the most common ways for a user to communicate with a computer in a dialogue-based system is through a microphone or through a telephone, 116 connected by a telephone switching system, 114 to a computer on which the software enabling the human and computer to interact is stored in a database, 112. Each interaction between the computer and the user in which the computer tries to elicit a

particular piece of information from the user is called a state or a turn. In each state the computer starts with a prompt and the user gives a spoken response. The application must recognize and interpret what the user has said, perform the appropriate action based on that response and then move the conversation to the next state or turn. The steps are as follows:

1. The computer issues a prompt.
2. The user (or caller) responds
3. The speech recognizer converts the response to computer-readable form.
4. The application interprets the response and acts accordingly. This may involve data base access for a query, for example.
5. The application may respond to the user.
6. Steps 1 through 5 may be repeated until a satisfactory response is received from the user.
7. The application transitions to the next state.

Hence a dialogue-based speech application includes a set of states that guide a user to his goal. Previously the developer had to code each step in the dialogue, coding for each possible event and each possible response in the universe of possible events, a time-consuming and technically-complex task. The developer had to choose an interactive voice response (IVR) system, such as Parity, for example, and code the application in the programming language associated with that language, using a speech recognition engine such as Nuance, Lernout and Hauspie or another speech recognition engine that would plug into the IVR environment.

Speech objects are commercially available. Referring to Fig. 3, speech objects, 322, 324 are pre-packaged bits of all the things that go into a speech act, typically, a prompt, a grammar, and a response. In this scheme, a speech object, for example, Get Social Security Number, 322, is purchased from a vendor. A developer writes software code, 320, in the programming language required for the speech objects chosen, and places the purchased Get Social Security Number speech object, 322, into his software. When the program executes and reaches a point where the social security number is required, the Get Social Security Number speech object, 322, is invoked. The application may have changed slightly how the question was asked, but the range of flexibility of the

speech object is limited. After the response from the user is obtained, control is returned to the application, 320. The application, 320, written by the developer, then must handle the transition to the next state, Get PIN Number, 324, and so on. Speech objects are implemented to a specific deployment system (e.g. Nuance's "IVR system" called Speech Channels, and SpeechWorks' "IVR system" referred to as an application framework). These reusable pieces are only reusable within the environment for which they were built. For example, a SpeechWorks implementation of this, called Dialog Modules, will only work within the SpeechWorks application framework.) The core logic is not reusable because it is tied to the implementation platform.

DFI Design Tool

In contrast, in accordance with the present invention, referring to Fig. 4, the developer would use the DFI design tool, 400, to enter a design of the whole application, as depicted in step 410, including many such states such as Get Social Security Number, Get PIN Number and so on. Once the application is rehearsed in the simulator (see reference patent Serial No. 09/417,166), step 420, files may be generated that represent that design, steps 440 and 450.

As shown in Fig. 5, the software application, 510, coded by the developer in any of a variety of programming languages, instantiates the dialogue flow interpreter, 530, and tells it to interpret the design specified in the file, 520, generated above by the DFI design tool. The dialogue flow interpreter, 530, controls the flow through the application, supplying all the underlying code, 540, that previously the developer would have had to write.

As can be seen from Fig. 6A, 612 and Fig. 6B, 622, the amount of code having to be written by a programmer is substantially reduced. Indeed, in some applications it can be entirely eliminated.

Dialogue Flow Interpreter

The Dialogue Flow Interpreter, or DFI, of the present invention provides a library of "standardized" objects that implement low-level details of dialogues. The DFI may be implemented as an application programming interface (API) that simplifies the implementation of speech applications. The speech applications may be designed using a

tool referred to as the DFI Development Tool. The simplification provided by the invention comes from the fact that the DFI is able to drive the entire dialogue of a speech application from start to finish automatically, thus eliminating the crucial and often complex task of dialogue management. Traditionally, such a process is application dependent and therefore requires re-implementation for each new application. The DFI solves this problem by providing a write-once, run-many approach.

Figure 2 illustrates the relationship between the DFI Design Tool, 210, the Dialogue Flow Interpreter, 232, and other speech application components. (In this diagram, block arrows illustrate the direction of data flow.)

Functional Elements

A speech application includes a series of transitions between states. Each state has its own set of properties that include the prompt to be played, the speech recognizer's grammar to be loaded (to listen for what the user of the voice system might say), the reply to a caller's response, and actions to take based on each response. The DFI keeps track of the state of the dialogue at any given time throughout the life of the application, and exposes functions to access state properties.

Referring to Figure 7, it can be seen that state properties are stored in objects called "shared objects", 710. Examples of these objects include but are not limited to, a Prompt object, a Snippet object, a Grammar object, a Response object, an Action object, and a Variable object.

Exemplary DFI functions, 780, return some of the objects described above. These functions include:

GET-PROMPT, 720: Returns the appropriate prompt to play. This prompt is then passed to the appropriate sound playing routine for sound output.

GET GRAMMAR, 730: Returns the appropriate grammar for the current state. This grammar is then loaded into the speech recognition engine.

GET RESPONSE, 740: Returns a response object comprised of the actual user response, any variables that this response may contain, and all possible actions defined for this response

ADVANCE-STATE, 750: Transitions the dialogue to the next state.

Other DFI functions are used to retrieve state-independent properties (i.e., global project properties). These include but are not limited to:

Project's path, 760

Project's sounds path

Input Mode (DTMF or Voice)

Barge-in Mode (DTMF or Voice)

Current State

Previous State

DFI Alternative Uses

Logging device for dialogue metrics – Because the DFI controls the internals of transitioning between states, it would be a simple matter to count how many times a certain state was entered, for example, so that statistics concerning how a speech application is used or how a speech application operates, may be collected.

- Speech application stress tester – Because the DFI controls the internals of transitioning between states, the DFI Tool enables the development of a application (using text to speech) that would facilitate the testing of speech applications by providing the human side of the dialogue in addition to the computer-side of the dialogue.

Figure 7 illustrates how the DFI functions 780 may be implemented or viewed as an applications programming interface (API).

Comparison of DFI to Speech Objects

Speech Objects (a common concept in the industry) represent prepackaged bits of all the things that go into a "speech act," typically, a prompt (something to say), a grammar (something to listen for) and perhaps some sort of reaction on the part of the system. This might cover the gathering of a single bit of information (which seems simple until you consider everything that could go wrong). One approach is to offer pre-packaged functionally (e.g., SpeechWorks (www.speechworks.com)). An example of the basic model is as follows: The designer buys (e.g., from Nuance) a speech object called Get Social Security Number and puts it into his program. When the program reaches a

point where a user's social security number is needed, the designer invokes the Get Social Security Number object. The application may have altered it a bit by changing exactly how the question is asked or extending the range of what it will hear, but the basic value is the prepackaged methodology and pre-tuned functionality of the object.

In the Dialogue Flow Interpreter Development Tool of the present invention, the designer would use a design tool (say, the DFI tool offered by Unisys Corp.) to enter a design of the whole application (potentially including many states such as getting SS# and getting PIN and so on). Once this application is rehearsed in a simulator (Wizard of Oz tester), files are generated that represent that design (e.g., MySpeechApp). The DFI is instantiated by the "runtime" application (written in some programming language) and told to interpret the design (MySpeechApp) produced by the design tool. Once set up, the application code need only give the DFI the details of what is going on to "read back" the design for what to do next. So, for example, the designer may indicate a sequence such as:

```
What is your SS Number?
(listen for SS Number)
What is your PIN
(listen for PIN)
Do you want to order or report a problem
(listen for ORDER or REPORT_A_PROBLEM)
if ORDER then
        What is your order...
else if REPORT A PROBLEM then
        What is your problem...
```

In this case, the DFI would first enter a state where, when the program asked what prompt to play, it would return "What is your SS Number?," and indicate that the program should listen for the SS#. Once the application told the DFI this had been accomplished and to move on, the application would again ask the DFI what to say and it would now return "What is your PIN". The DFI would continue supplying directional data until the application ended. The point is that the DFI supplies the "internals" for each

turn of the dialogue (prompt, what to listen for, etc) as well as the flow through the application.

Although they address similar problems, the DFI is very different from of the Speech Objects model. Speech Objects set up defaults a program can override (the program has to know this from somewhere) whereas DFI provides the application with what to do next. Speech Objects are rigid and preprogrammed and of limited scope, whereas the DFI is built for a whole application and is dynamic. Speech Objects are "tuned" for a special purpose. The aforementioned This tuning may be provided through the DFI design tool, as well. Another way to think of the difference is that the DFI delivers "custom" speech capabilities built through the tool, including how they "link" together. Speech Objects provide "prepackaged" capabilities (with the advantage of "expert design" and tuning) and with no "flow" between them.

Translator Object Classes

A speech application needs to be able to retrieve information in a form that the software can interpret. Once the information is obtained, it may be desirable to output that information in a particular speech format to the outside world. In accordance with the present invention, translator object classes enable a developer to provide parameters to specify details about how a particular piece of information should be output and the DFI will return everything necessary to perform that task. For example, when the desired object is to output what time it is presently in Belgium in English in standard time, the developer would specify the language (English), the region (Belgium), the time (the time right now in Belgium) and the format (standard time), and the DFI will return a play list of everything required to enable the listener to hear the data structure with those characteristics (the time in Belgium right now in standard format, spoken in English.)

For example, when the DFI is completing the prompting, the DFI would access the function GET PROMPT, Fig. 7, 720, which would return, (when the output speech is a recorded file):

1. the "It is now".wav file,
2. the value of the time instance (variable), 12:35pm: and the associated files:

twelve.wav

thirty.wav

five.wav

pm.wav,

3. and the "in Belgium".wav file.

The listener would hear: "It is now twelve thirty-five pm in Belgium." It should be understood that the above example is for exemplary purposes only. The present invention also includes text-to-speech (computer-generated) speech output.

Alternately, if the developer wanted to use the object directly in his application, without using the DFI, the application could access the translator directly. The translator would return the value of the time instance (12:35) and the associated files:

twelve.wav

thirty.wav

five.wav

pm.wav. Thus the translator object classes contain objects that can be used by the speech application written by the developer or by the DFI.

Although commercially available speech objects may provide similar functionality, the inventiveness of translator object classes lies in that the developer does not lose control of the low-level details of the way the information is output because the developer can write his own objects to add to the class. When a developer uses commercially available speech objects, the developer must accept the loss of flexibility to control the way the speech object works. With translator objects according to the present invention, the developer maintains control of the low-level details while still obtaining the maximum amount of automation.

Conclusion

In sum, the present invention provides system and methods to create interactive dialogues between a human and a computer, such as in an IVR system or the like. It is understood, however, that the invention is susceptible to various modifications and

alternative constructions. There is no intention to limit the invention to the specific constructions described herein. On the contrary, the invention is intended to cover all modifications, alternative constructions, and equivalents falling within the scope and spirit of the invention. For example, the present invention may support non-speech-enabled applications in which a computer and a human interact. The present invention will allow the recall of a textual description of a prompt which may be displayed textually, the user responding by typing into an edit box. In other words, it is the dialogue flow and properties of each state that is the core of the invention, not the realization of the dialog. Such an embodiment may be utilized in a computer game or within software that collects configuration information, or in an Internet application which is more interactive than simple graphical user interface (GUI) techniques enable.

It should also be noted that the present invention may be implemented in a variety of computer environments. For example, the present invention may be implemented in Java, enabling direct access from any Java programming language. Additionally, the implementation may be wrapped by a COM layer, allowing any language which supports COM to access the functions, thus enabling traditional development environments such as Visual Basic, C/C++, etc. to use the present invention. The present invention may also be accessible from inside Microsoft applications, including but not limited to Word, Excel, etc. through, for example, Visual Basic for Applications (VBA). Traditional DTMF-oriented systems, such as Parity, for example, which are commercially available, may embed the present invention into their platform. The present invention and its related objects may also be deployed in development environments for the world wide web and Internet, enabling hypertext markup language (HTML) and similar protocols to access the DFI development tool and its objects.

The various techniques described herein may be implemented in hardware or software, or a combination of both. Preferably, the techniques are implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program code is applied to data entered using the input device to perform the functions described above and to generate output information. The output information is applied to

one or more output devices. Each program is preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language. Each such computer program is preferably stored on a storage medium or device (e.g., ROM or magnetic disk) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described above. The system may also be considered to be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Although an exemplary implementation of the invention has been described in detail above, those skilled in the art will readily appreciate that many additional modifications are possible in the exemplary embodiments without materially departing from the novel teachings and advantages of the invention. Accordingly, these and all such modifications are intended to be included within the scope of this invention.

CLAIMS

We claim:

1. A method of developing a dialogue-enabled application for executing on a computer that enables a human and a computer to interact, comprising the acts of:

   (a) inputting instructions specifying the flow of a conversation to a design tool, said design tool producing a data file, said data file containing information concerning prompts, responses, branches and conversation flow for implementing a human-computer speech-enabled interaction; and

   (b) instantiating an interpreter object within the application, the interpreter object interpreting the data file to provide the human-computer dialogue-enabled interaction defined by the data file.

2. The method of claim 1 wherein said data file further contains information concerning a speech recognition engine.

3. The method of claim 1 wherein said data file is automatically stored.

4. The method of claim 1 wherein said inputting of instruction takes place through a graphical interface.

5. A system for developing dialogue-enabled software for executing on a computer that enables a human and a computer to interact comprising:

   a design tool for accepting instructions specifying the flow of a conversation, said design tool producing a data file; and

   an interpreter for interpreting said data file, said interpreter automatically enabling the human-computer interaction.

6. The system of claim 5 further comprising a library, wherein the library contains said data files.

7. The system of claim 5, wherein the design tool further comprises a graphical interface.

8. A computer-readable medium comprising computer executable instructions for instructing a computer to perform the acts of:
accepting instructions, said instructions specifying a flow of conversation between a human and a computer;
producing a data file for input to an interpreter;
interpreting said data file; and
providing the human-computer dialogue-enabled interaction.

9. The computer-readable medium of claim 8 containing further instructions enabling the generated code to be immediately accessible to other software developers.

10. A dialogue flow interpreter (DFI) for use in computer-implemented system for carrying out a dialogue between a human and a computer, wherein the DFI comprises computer executable instructions for reading a data file containing information concerning prompts, responses, branches and conversation flow for implementing a human-computer dialogue, and computer executable code for using said information in combination with a library of shared objects to conduct said dialogue.

11. A DFI as recited in claim 10, wherein the DFI is implemented in an application comprising, in addition to the DFI, a language interpreter, recognition engine, and voice input/output device.

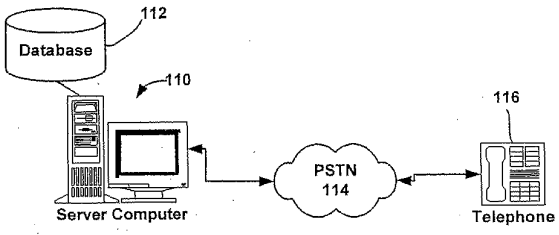

Figure 1 (Prior Art)

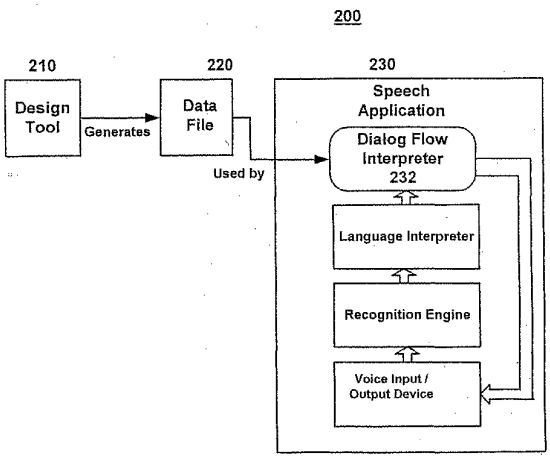

Figure 2
DFI Development Tool

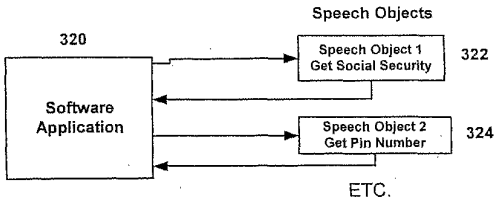

Figure 3 - Prior Art

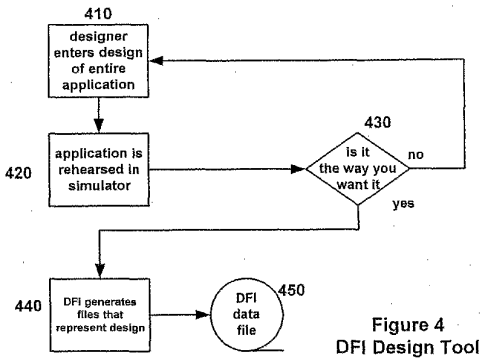

Figure 4
DFI Design Tool

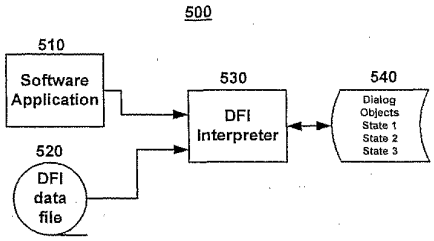

Figure 5
DFI

Figure 6(A) - Prior Art

Figure 6(B)
DFI Tool

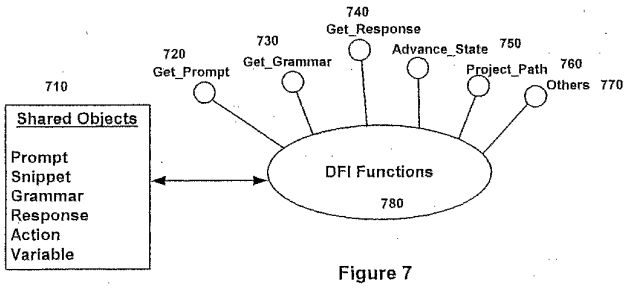

Figure 7
DFI Functions

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 May 2002 (10.05.2002)

PCT

(10) International Publication Number
WO 02/037268 A3

(51) International Patent Classification⁷: G06F 9/44

(21) International Application Number: PCT/US01/50119

(22) International Filing Date: 19 October 2001 (19.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/702,224 31 October 2000 (31.10.2000) US

(71) Applicant: UNISYS CORPORATION [US/US]; Township Line and Union Meeting Roads, P.O. Box 500, Blue Bell, PA 19424-0001 (US).

(72) Inventors: SCHOLZ, Karl, Wilmer; 203 Vassar Circle, Strafford, PA 19807 (US). IRWIN, James, S.; 55 Ream Road, Stevens, PA 17578 (US). TAMRI, Samir; 333 Lancaster Avenue, #309, Frazer, PA 19355 (US).

(74) Agents: STARR, Mark, T. et al.; Unisys Corporation, Township Line and Union Meeting Roads, P.O. Box 500, Blue Bell, PA 19424-0001 (US).

(81) Designated States *(national)*: CA, JP.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
with international search report

(88) Date of publication of the international search report:
31 July 2003

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/037268 A3

(54) Title: DIALOGUE FLOW INTERPRETER DEVELOPMENT TOOL

(57) Abstract: A computer software product (200) is used to create applications for enabling a dialogue between a human and a computer. The software product provides a programming tool (210) that insulates software developers from time-consuming, technically-challenging programming tasks by enabling the developer to specify generalized instructions to a Dialogue Flow Interpreter (232), which invokes functions to implement a speech application, automatically populating a library (540) with dialogue objects that are available to other applications. The speech applications created through the DFI may be implemented as COM (component object model) objects, and so the applications can be easily integrated into a variety of different platforms. In addition, "translator" object classes are provided to handle specific types of data, such as currency, numeric data, dates, times, string variables, etc. These translator object classes have utility either as part of the DFI library or as a sub-library separate from dialogue implementation.

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Interna application No
PCT/US 01/50119

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	KIMURA T D ET AL: "FORM/FORMULA FORM/FORMULA A VISUAL PROGRAMMING PARADIGM FOR USER-DEFINABLE USER INTERFACES A VISULA PROGRAMMING PARADIGM FOR USER-DEFINABLE USER INTERFACES" COMPUTER, IEEE COMPUTER SOCIETY, LONG BEACH., CA, US, US, vol. 28, no. 3, 1 March 1995 (1995-03-01), pages 27-35, XP000510127 ISSN: 0018-9162 the whole document --- -/--	1-11

[X] Further documents are listed in the continuation of box C. [ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 April 2003	28/04/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Brandt, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Interna Application No
PCT/US 01/50119

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	FENG-YANG KUO ET AL: "USER INTERFACE DESIGN FROM A REAL TIME PERSPECTIVE" COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY, ASSOCIATION FOR COMPUTING MACHINERY. NEW YORK, US, vol. 31, no. 12, 1 December 1988 (1988-12-01), pages 1456-1466, XP000120403 ISSN: 0001-0782 the whole document	1-11

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

フロントページの続き

(74)代理人 100096781
弁理士 堀井 豊
(74)代理人 100098316
弁理士 野田 久登
(74)代理人 100109162
弁理士 酒井 將行
(72)発明者 ショルツ，カール・ウィルマー
アメリカ合衆国、１９８０７ ペンシルバニア州、ストラフォード、バッサー・サークル、２０３
(72)発明者 アーウィン，ジェイムズ・エス
アメリカ合衆国、１７５７８ ペンシルバニア州、スティーブンス、リーム・ロード、５５
(72)発明者 タムリ，サミール
アメリカ合衆国、１９３５５ ペンシルバニア州、フレイザー、ランカスター・アベニュ、３３３、ナンバー・３０９
Ｆターム(参考) 5B076 DC01 DF07