1U－2

マルチモーダル・インタフェース構築のためのソフトウェアツール

川合 史朗，相田 仁，齊藤 忠夫
東京大学 工学部

1 はじめに

文字情報だけでなく画像情報や音声情報など、複数のモダリティを利用したアプリケーションは、操作性、作業効率、学習のしやすさ等の面で格段に優れている。しかしその一方で、情報が多くのコミュニケーションチャネルに分散して伝達されるようになってくると、そのうちひとつでも利用できないコミュニケーションチャネルがあるユーザはアプリケーション全体を利用できなくなるという問題が生じてきた[1]。

例えば、キャラクタベースのアプリケーションならば、全盲の人でも画面に表示されたキャラクタを音声や点字ディスプレイに出力する付加機器によって利用することができたが、ビットマップで出力されるグラフィックベースのアプリケーションではそのような方法は使えない。

そのようなケースに対し、特別なインタフェースを用意するのではなく、ユーザの特性や使用環境からくる要請に対し、その都度適したモダリティを用いることができるようにアプリケーションを設計することを容易にするツールキットを提案する。

2 ツールキットベースアプローチ

アプリケーションから提示される情報をどのレベルで切り替えるかという選択はいくつか考えられる (図1)。

OSレベルで複数のモダリティに対応することができれば、アプリケーションから独立した汎用的な選択機構が実現できる[2]。また入力に困難がある場合にはデバイスドライバレベルでの対応で充分な場合も多い[3]。

しかし、GUIを用いたアプリケーションの出力のセマンティクスを従来のOSの段階で捕捉することは非常に困難であり、OSのAPIとしてより高度な表現を採用するか、ツールキットレベルで対応する必要があると考えられる。

すなわち、図2のようにGUI、キャラクタベースユーザインタフェース (CUI)、音響的ユーザインタフェース (AUI) 等のツールキットとアプリケーション間のインタフェースを一様にしておき、ユーザの状況や使用環境に応じて必要なモダリティを切り替えて使えるようにするのである。

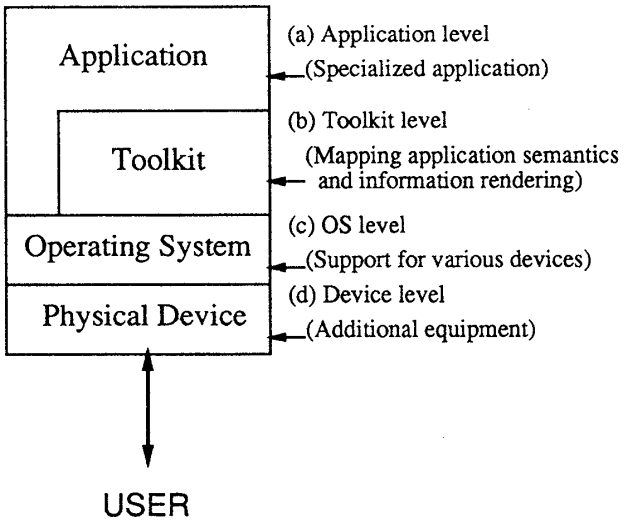

図 1: 代替コミュニケーションを用意するレベル

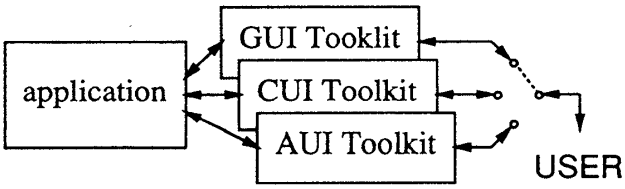

図 2: ツールキットベースアプローチ

3 モーダル独立なアプリケーション記述

3.1 Abstract widget

上記のようなアプローチを容易にするためには、汎用的なツールキット・アプリケーション間インタフェースを決める必要がある。

従来のGUI用ツールキットは、視覚的にまとまった部品 (widget) を中心として組み立てられており、これをそのまま他のモダリティに拡張するのにはやや難がある。ジオメトリ情報はCUIやAUIでは全く必要ない。一方、GUIではオブジェクトの状態は画面にいつも表示されているが、AUIでは、音響情報が一過性のものであるという性質上、ユーザが必要なときにオブジェクトの状態を問い合わせるプロトコルを用意しておかなければならない。

したがって、インタフェースモードに独立にアプリケー

" A Toolkit for Developping Multimodal Interface"
Shiro Kawai, Hitoshi Aida and Tadao Saito
Faculty of Engineering, The University of Tokyo

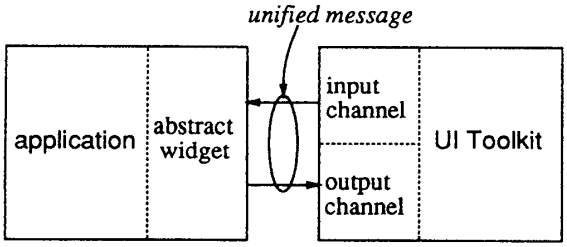

図 3: Abstract widget

ションを書くには、GUI ツールキットよりも抽象的なレベルでオブジェクトを用意してやる必要がある。これを abstract widget と呼ぶことにする (図 3)。

ユーザからの入力は、まずモダリティ毎のユーザインタフェースツールキットによって、モダリティに依存した形式からモダリティに依存しない abstract widget へのメッセージという形に変換される。このマッピングを行なう部分を input channel と呼ぶ。Abstract widget はメッセージを受け取ると、その定義に従って内部情報の更新、他の abstract widget とのメッセージ交換等を行い、必要があれば output channel を通してユーザに情報を送る。output channel は abstract widget からの出力 (モダリティ独立) を利用可能なモダリティでの表現にマップする。

例えば、アプリケーションを終了させるというインタラクションを考える。GUI では、"Quit" というプッシュボタンをマウスでクリックすることがアプリケーションを終了させるかもしれない。一方、キャラクタオリエンテッドなアプリケーションでは、キーボードから 'q' あるいは "quit<RET>" と入力することにより終了するかもしれない。この場合、input channel は与えられたコンテキストからマウス入力あるいはキーボード入力を "activate" メッセージにマップし、そのメッセージを「アプリケーションを終了させる」という abstract widget に送る。

3.2 データの取扱い

ユーザは、情報を検索・入手し、あるいは創作・加工・発信することを目的として計算機を用いる。その情報そのものを原情報と呼ぶことにする。原情報には、テキスト (単なる文字情報、あるいはフォーマットされたもの)、静止画像、動画像、音響データ、およびそれらが同期して関連づけられたものなどが考えられるが、それぞれに表現に適した "自然なモダリティ" を持っている。"自然なモダリティ" はその情報を表現するために本質的なものであり、そのモダリティが使えない場合は何らかの代用表現を用いるしかない。

原情報そのものとその代用表現とをパッケージ化したものをマルチモーダルデータオブジェクト (MDO) と呼ぶ。MDO を生成するには、原情報に代用表現を付加する (画像データに対しその説明をテキストで付加するといった場合) か、原情報から代用表現を生成するメソッドを付加 (テキストデータに対し、それを音声化するためのメソッドを付加するといった場合) すればよい。

Abstract widget と output channel 間で MDO をやり取りすることにより、abstract widget の記述に関係なく、output channel は要求されたモダリティでデータを表現することができる。

Input/output channel 及び abstract widget の動作は原則として非同期である。出力するのにある程度の時間幅を要するような MDO を出力した場合でも、abstract widget からはすぐに制御が戻る。

3.3 Abstract Widget の記述

プロトタイプは Scheme を使用して作られており、abstract widget も Scheme の拡張構文によって記述される。各 abstract widget は、入力メッセージ − アクション − 出力のルールの記述から、first class object として生成される。

現在、オブジェクトオリエンテッドプログラミングのサポートを組み込んだ Scheme を用いて、UNIX WS 上でプロトタイプシステムを開発しながら、仕様の詳細化を行なっている。

4 おわりに

複数のモダリティを柔軟に切替えて使うことができるためのツールキットについて述べた。

他のモダリティのサポートをアプリケーションに組み入れることによって開発コストが大幅にはねあがってしまうようでは、多くのアプリケーションベンダがサポートを避けてしまうだろう。より使いやすいアプリケーションが作られるためには、そのようなアプリケーションの開発を容易にする環境の構築も重要であると考える。

参考文献

[1] G. C. Vanderheiden. Building Disability Access Directly into Next-Generation Information and Transaction Systems. In *Proceedings of the IISF/ACMJ International Symposium on Computers as Our Better Partners*, pages 2–6. World Scientific, Mar. 1994.

[2] 小山 智史, 野島 秀夫, and 太田 茂. 盲人のためのコンピュータインタフェース —OS-TALK—. 電子情報通信学会創立 *70* 周年記念総合全国大会, S13-11, 1987.

[3] 太田 茂. 障害者の可能性を広げるコンピュータ. 中央法規出版, 1990.