ヒューマンインタフェース 59−10
（1995. 3. 3）

ユーザレベルでカスタマイズ可能なGUI部品の交換・付加機能

佐藤 博之, 増田 英孝, 笠原 宏 (東京電機大学 工学部)

概要

我々は、グラフィカルユーザインタフェース (GUI) の作成と実行の境界を無くし、ユーザがGUIに対して徹底的に手が加えられるような環境の研究を行っている。アプリケーション本体に手を加えることなく、同等の機能を持つGUI部品同士の交換および補助機能を持つ部品の付加を可能とする方法について検討している。

本稿は、エンドユーザによるアプリケーション実行時にGUI部品の交換を可能とするASIA (Available and Selectable Interface Adaptor) を提案するものである。例えば単一選択を行うという同等の機能を持つ、リストとラジオボタン群の交換について検討した。またポップアップメニューのコマンド実行を、アクションボタンとして付加することも検討した。これによりエンドユーザは、数種の利用可能なGUIの中から個人の好みに応じたGUIの選択が可能となる。

A Customize Function on User's Level to Exchange and Add GUI Components

Hiroyuki SATO, Hidetaka MASUDA, Hiroshi KASAHARA
Department of Electrical Engineering, Faculty of Engineering, Tokyo Denki University,
2-2 Kandanishiki-cho, Chiyoda-ku, Tokyo 101, Japan

Abstract

We have researched a platform that allows users to thoroughly modify GUI (Graphical User Interface), by means of an elimination of the barrier between GUI *development* and GUI *running*. We have researched the way to exchange the GUI component for another same functional component.

We propose *ASIA (Available and Selectable Interface Adaptor)* that allows users to exchange GUI components during application runtime. For example, to exchange a single selection list for radio buttons. So a user can select one's favorite GUI components, and customize the application's user interface.

1 はじめに

GUIを使用したアプリケーションは、アプリケーション本体とUI (User Interface) を分離する構成で設計されている。そのため同一のアプリケーションであっても、様々なUIが接続可能となっている。しかしUIの選択はアプリケーション設計段階までが現状で、ユーザ (オペレータ) がアプリケーション使用の段階で好みに応じて選択できるわけではない。

アプリケーション設計段階で、アプリケーション本体に自由にGUI部品を選択できるようになっていても、完成されたアプリケーションは、実行段階ではそのGUI部品と密接に結合してしまっている。

本研究の目的は、プログラミング知識のないユーザによって、アプリケーションのGUI部品の自由な交換・付加を可能とし、アプリケーションのGUIを実行時にカスタマイズ [1] できるようにすることである。本研究で提案するASIA (Available and Selectable Interface Adaptor) [2] は、GUI部品を複数種類生成し必要な部品のみを表示することによって、ユーザがGUI部品を交換または付加することができる。また部品構成が動的に変更可能となるので、ユーザに対して数種の利用可能なGUIの中から、個人の好みに応じたGUIの選択を可能とするものである。

2 ユーザインタフェースの柔軟性

現在では、コンピュータの専門家ではない多くの人達が、コンピュータを利用する状況がますます増えている。この状況の中で必然的に、人とコンピュータとの橋渡しとなるユーザインタフェースが重要視されている。

ユーザがコンピュータを使って快適に作業できるために、ユーザインタフェースが備えるべき性質として次の3つが挙げられている [3]。

1. 一貫性 (consistency)
2. 柔軟性 (flexibility)
3. 協調性 (cooperativeness)

大多数のユーザに共通して利用できるためには一貫性が重要である。

しかし本研究では、ユーザの好みに応じてGUI部品を交換する機能を提供することで、上記の内の柔軟性をより向上させることを目的としている。

3 GUI部品交換の問題点

アプリケーション本体とUIを分離することにより、図1のように同一のアプリケーション本体に対して複数のUI戦略が利用可能となっている。図1でUI, UI1, UI2は同等の機能を持つ部品である。しかし複数のUIの中からある1つを選択するのは設計者であり、複数のUI戦略が利用可能という利点は、アプリケーションの設計段階までしか活用されていないのが現状である。ユーザの好みに応じたGUIにカスタマイズできるようにするためには、この利点をアプリケーション実行時にも活用することである。アプリケーション実行時にGUI部品の交換が可能となれば、ユーザに対しより柔軟性のあるUIを提供できることになる。

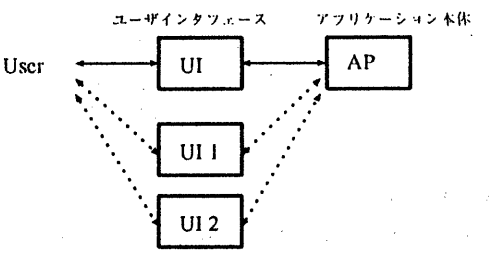

図1: 同一のアプリケーション本体に対する複数のUIの選択

これを実現するために現在のアプリケーション構成の問題点として、次のようなものが考えられる。

- 実行時におけるアプリケーション本体とUI部の接続のハードカップリング
- アプリケーション本体とUI部の接続インタフェースの不整合 [4]

接続のハードカップリングは、実行時にはアプリケーション本体とUI部が密接に結合していなければ、アプリケーションとして動作しないということである。完成されたアプリケーションは、その本体とUI部を含めた形で、1つのアプリケーションとなっている。そこで我々は、アプリケーション実行時にUIを自由に着脱できる機能を提案する。

一方の接続インタフェースの不整合は、ユーザにとって交換可能と思われる部品でも、アプリケーション本体とのAPI (Application Interface)が異なっているので簡単には交換できないことである。例えばユーザにとってどちらも単一選択を行うための部品である、単一選択リストとラジオボタン群は交換可能と考えられる。図2と図3にその例を示す。リストは単体部品である一方、ボタン群は複数の部品から構成されている。図4と図5のようにAPIが異なっているので、このままでは交換できないことがわかる。

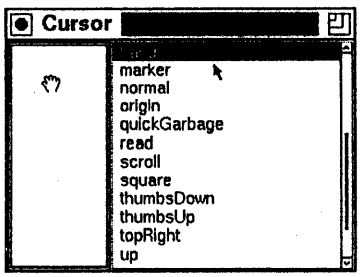

図2: リストを用いたアプリケーションの例

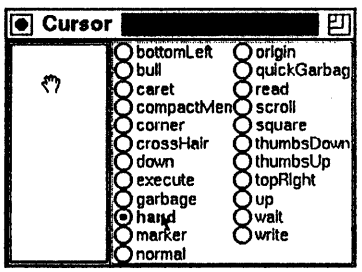

図3: ラジオボタンを用いたアプリケーションの例

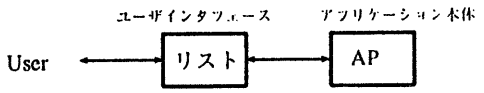

図4: アプリケーション本体とリストの接続図

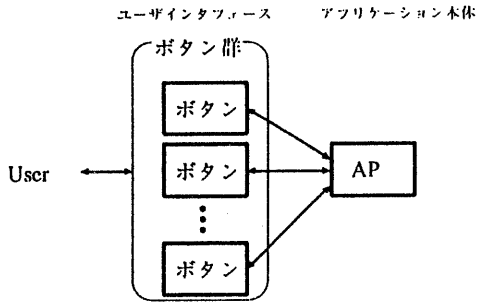

図5: アプリケーション本体とボタンの接続図

4 ASIAの構想

実際にGUIの部品構成を変更しようとすると、プログラミング知識のないユーザにとっては不可能なことである。アプリケーションプログラマにとっても、特定のアプリケーションだけに対する変更と限定されてしまうのは非効率的である。GUI部品の交換は、アプリケーションに依存せず、GUI側で対応すべきであるということがASIAの基本構想である。アプリケーション本体に直接GUI部品を接続する代わりに、ASIAがアプリケーションとGUI部品の間に入り、複数の部品の生成を行い、必要な部品を表示する。

GUI部品構成の変更には、同一機能を持つ部品同士の交換以外に、ある部品に対する機能の拡張や支援をする部品を付加することもある。ASIAではこれらも、ある部品とその他の部品が付加された部品群との交換と捉え、

- 部品と部品の交換
- 部品の付加

の2つの機能を併せ持っている。

ASIAは交換し得るGUIを、ユーザがアプリケーション実行時に自由に選択できることを目的

としている。また既存のアプリケーションのGUIにもASIAを適用すれば、それらの部品交換が可能となる。

これらを実現するために、ASIAはユーザが選択した交換の対象となる部品を保持し、その部品から別の部品を生成する機能を持っている。部品を生成する新たなメソッドをプログラマが追加すれば、そのASIAを使っているアプリケーションに別の新しいGUIを提供できる。ASIAは、この部品生成機能から作られた部品を交換するため、それら全部品をコンポーネントとするコンポジット(CompositePart)である。そしてコンポーネントのレイアウト(表示領域)を変更する機能を持たせることで、動的な部品の交換が可能となる。

ユーザがある部品に対し交換や付加のカスタマイズをしたい時は、Metamer [1]の機能を利用してその部品を選択し、ASIAの機能を呼び出す。通常のアプリケーションは、図6のようにUIと接続されている。部品を交換・付加するためには、まずその部品とアプリケーション本体の間にASIAを挿入する。挿入されたASIAは交換あるいは付加できる部品を生成する。初期設定では元の部品のみを表示する。そしてユーザの指示によって必要な部品だけを表示する。例えば図7で部品1に部品2を付加する機能が呼び出されたら、部品2を表示し、それにより占有された分だけ部品1のレイアウトを縮小する。ユーザは図8のように部品1と部品2を使用することができるようになる。

ASIAは、

- あるGUI部品に対し交換可能なGUI部品を複数種類、生成可能
- ユーザによる、アプリケーション実行時のGUI部品交換・付加が可能
- 既存のアプリケーションのGUIにも適用可能

という特長を持つ。GhoustHouse [5]やTUIMS [6]など、部品交換・付加機能を持つクラスライブラリのGUIカスタマイズ機能では、新たに作成したアプリケーションにしか利用できない。

5 ASIAの実装

我々はASIAを*Objectworks\Smalltalk* Release 4.1 [7]上に実装した。既存のアプリケーションに使用されているGUI部品にも、ASIAを適用できるようにすることを目的として設計した。部品(コンポーネント)の交換・付加は、まず必要な部品をすべて生成し、各部品のレイアウトを変更することで実現している。不要の部品はレイアウトを未定義(nil)とすることで、ユーザからは見えなくすることが可能である。

5.1 抽象クラスASIA

ASIAは複数のコンポーネントを持つことが可能で、それらを動的に組み換えられるものである。CompositeViewのサブクラスとして抽象クラスASIAを作成した。CompositeViewはCompositePartのサブクラスであるので複数のコンポーネントを持つことができ、またViewのようにControllerを持つことができる。ASIAはこれらの機能を継承している。

抽象クラスASIAは、自分のコンポーネントをリセット(再配置)する機能(メソッド)を持っている。従って各サブクラスとなる具象ASIAは、「コンポーネントをリセットせよ(anASIA resetComponents)」というメッセージを受信した時に、このメソッドを利用することができる。各ASIAが複数のコンポーネントを保持するために、ASIAはインスタンス変数componentDictionaryを持っている。このインスタンス変数には、各コンポーネントをキーとし、それぞれのレイアウトを値として持つDictionaryが保持される。どんなコンポーネントを持ち、どんな配置にするかは、具象ASIAに委任されている。ASIAは「コンポーネントをリセットせよ」というメッセージを受信したら、一度自分のコンポーネントをすべて取り除く(図9)。そしてインスタンス変数componentDictionaryに保持されているレイアウトに従って、すべてのコンポーネントを配置する(図10)。レイアウトがnilならば、その部品は追加してもユーザには見えない。

ユーザが部品を交換する方法、つまりコンポーネントのレイアウトを変更する方法は、各ASIA

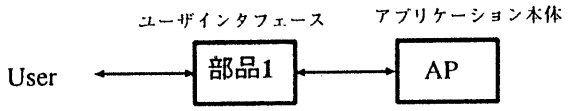

図 6: 通常のアプリケーション本体と GUI 部品の接続

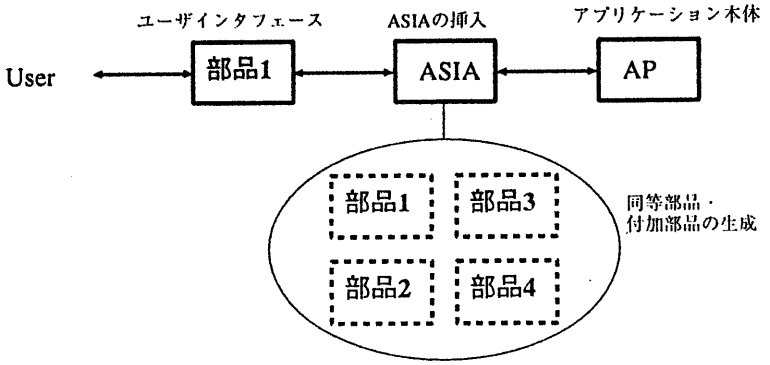

図 7: アプリケーション本体と GUI 部品間への ASIA の挿入と同等部品・付加部品の生成

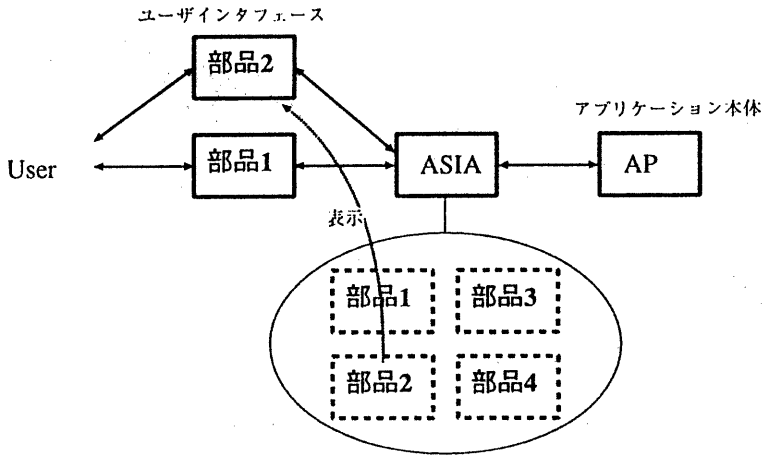

図 8: 必要な部品を表示することによる部品交換・付加機能

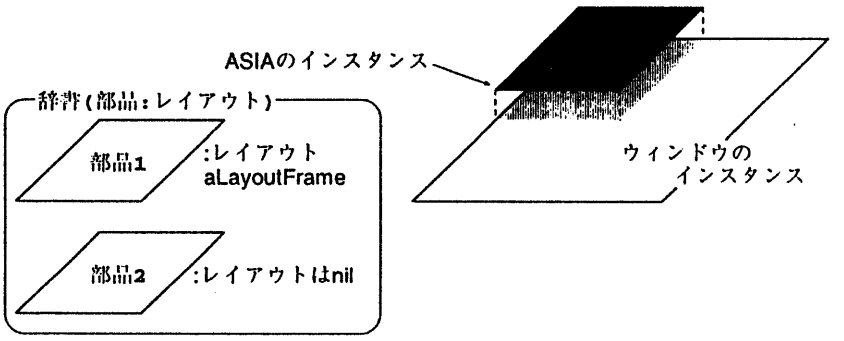

図 9: ASIA を挿入した場合のウィンドウをインスタンス構成

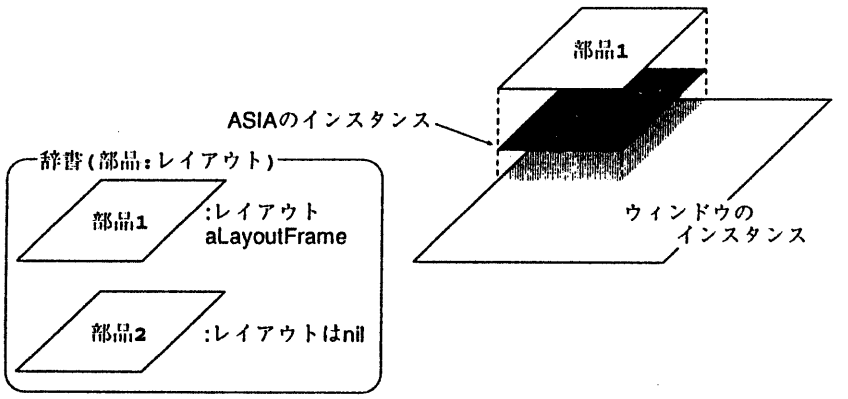

図 10: ユーザが必要とする部品の表示

に共通のコントローラであるASIAControllerからコンポーネントのレイアウトを変更するメッセージを送信する。メッセージを受信したそれぞれのASIAはレイアウトの変更を辞書に登録して、自分に「コンポーネントをリセットせよ」というメッセージを送信する。レイアウトの変更はそれぞれの具象ASIAに委任されている。

5.2 ポップアップメニューからアクションボタンを生成するASIA

MenuASIAは、ポップアップメニューを使用している部品に対し、そのメニューコマンドを実行するアクションボタンを生成する。インスタンス変数として、targetとactionButtonsを持っている。インスタンス生成時にtargetにBorderDecoratorを保持する。targetのコンポーネントとなっているオブジェクト(部品)からポップアップメニューを受け取る。そのポップアップメニューを使ってアクションボタン群を生成し、actionButtonsに保持させる。

図11と図12に、ファイルエディタにMenuASIAを適用した例を示す。図11に見られるポップアップメニューのコマンドを、図12のようにアクションボタンとして付加することができる。

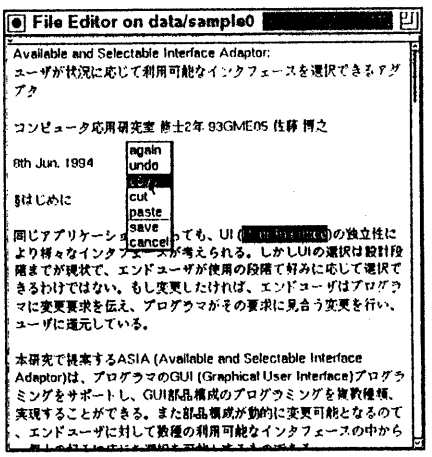

図11: MenuASIAの例: ポップアップメニューのみ使用可

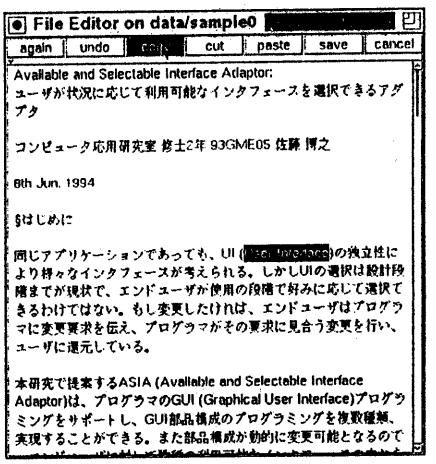

図12: MenuASIAの例: メニューのコマンドを実行するアクションボタンを付加

5.3 単一選択のためのASIA

SingleSelectionASIAは、単一選択リストに対し、ラジオボタン群を生成する。リストもポップアップメニューを使用しているので、MenuASIAのサブクラスとした。従って継承されるインスタンス変数targetとactionButtonsは、MenuASIAと同様なコンポーネントを保持する。またSingleSelectionASIAはインスタンス変数radioButtonsとupDownButtonsを持っている。targetのコンポーネントを使ってラジオボタン群を生成し、radioButtonsに保持させる。一方のupDownButtonsには選択されている項目を1つずつ前後に変更するボタンを生成し保持させる。

図2や図3のように、リストとラジオボタン群を交換することができる。また図13や図14のような、選択を変更するアクションボタンを付加することができる。

6 おわりに

アプリケーションのGUI部品構成を、アプリケーション実行時にユーザによって、カスタマイズ可能とする方法について検討した。本研究では、特にGUI部品の交換の機能をユーザに提供するこ

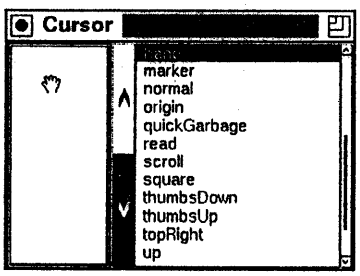

図 13: SingleSelectionASIA の例: リストに選択項目を変更するアクションボタンを付加

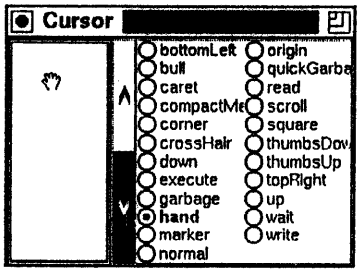

図 14: SingleSelectionASIA の例: ラジオボタンに選択項目を変更するアクションボタンを付加

とで、GUI のカスタマイズを可能とすることを目的とした。実際に GUI 部品交換をするための問題点を指摘し、それを解決するための ASIA を提案した。ASIA は、GUI 部品構成を複数種類生成することができ、アプリケーション実行時に動的に変更する機能も持っている。具体的なクラスとして、メニューを使っている部品を用いて、そのメニューのコマンドをアクションボタンに置き換えることができる MenuASIA と、単一選択を行う部品群を生成することができる SingleSelectionASIA を作成した。ASIA によって、ユーザは好みの GUI 部品を選択できる。今後は ASIA の評価実験や、その他の目的の部品に対する ASIA の検討を行う予定である。

謝辞

本論文をまとめるにあたって有益なアドバイスをして頂いた三菱電機株式会社情報システム研究所の宮崎一哉氏、株式会社 SRA プラクティカルソフトウェア工学研究所の渡邊克宏氏ならびに東京電機大学工学部情報通信工学科の寺島慎次教授に感謝致します。

参考文献

[1] 増田英孝, 笠原宏. アプリケーション実行時 GUI レイアウト変更機能. 情報処理学会論文誌, Vol. 35, No. 9, pp. 1794 1806, 1994.

[2] 佐藤博之, 増田英孝, 笠原宏. ASIA (Available and Selectable Interface Adaptor): ユーザが状況に応じて選択可能なインタフェースアダプタ. 第 49 回全国大会講演論文集, pp. 3N 9, 1994.

[3] 宮崎一哉. ユーザインタフェースに対する要求. 情報処理学会 夏のシンポジウム-ヒューマンフレンドリーなシステム, pp. 267 273, 1986.

[4] 佐藤博之, 増田英孝, 笠原宏. GUI 部品の交換によるデザイン変更のためのプラグインアダプタ. 第 48 回全国大会講演論文集, pp. 2J 2, 1994.

[5] 杉本明, 北村操代, 中田秀男, 川岸元彦, 小島泰三. 対話型システム視覚的構築用クラスライブラリ: GhostHouse (I) - 設計方針と概要 . 第 46 回全国大会講演論文集, pp. 6R 1, 1993.

[6] 立山義祐, 寺田実. 操作カスタマイズ機能を内蔵するツールキットの提案. 日本ソフトウェア科学会 WISS'94 インタラクティブシステムとソフトウェア II, pp. 105 114, 1994.

[7] ParcPlace Systems, Inc. *Objectworks\Smalltalk Release4.1 User's Guide*, 1992.