イベント-状態駆動GUI出力制御機構の設計と応用

Aiguo He[†1] 張 国珍[†2]
程 子学[†3] 小山明夫[†4]

仕様変更に柔軟に対応できるアプリケーションプログラムのGUI出力制御の設計と実装は経験の少ないプログラマにとって困難である．本論文が提案するGUI出力制御機構はアプリケーションの仕様変更に対する柔軟性を持ち，且つ簡単に実現できる特徴を有する．提案した方法は筆者らの遠隔教育支援システムの設計に応用され，システム開発効率の向上に寄与した．

The Design and Application of the Event-Condition Driven GUI Output Control Mechanism

AIGUO HE,[†1] GUOZHEN ZHANG,[†2] ZIXUE CHENG[†3]
and AKIO KOYAMA [†4]

For experienceless programmers it is difficult to design and develop the GUI output control which can well copes with the change of the specification of the application program. This paper proposes a GUI output control mechanism which can efficiently cope with the specification changes and can be achieved easily. The proposed mechanism was applied to the design of authors' distance education support system and contributed to the improvement of the system development efficiency.

1. はじめに

コンピュータハードウェアの高性能化に従い，Java[1]などのオブジェクト指向プログラミング言語によりイベント駆動モデルに基づいた複雑なグラフィカルユーザインタフェース（GUI）を有するソフトウェアシステムの開発が可能となった．またGUIの操作性向上を図るために，GUI出力の自動制御が一般的になってきている．GUI出力の自動制御とは，アプリケーションシステムの状態に応じて，GUI項目（ボタン，入力フィールド，文字列など）の属性（操作と入力の可能・不可能，表示される文字列の内容，色など）を自動的に制御することである．

GUIの自動制御が行われているシステムでは，例えばある機能を起動するボタンに対して常に可能の限りその機能が実行可能の状態にあるか否かをチェックし，チェックの結果に従ってそのボタンの操作可能か否かの属性を設定する．これにより利用者の無駄の操作を避けることができる．Microsoft社のOSソフトウェアWindows[2]はこのようなGUI出力自動制御が成功しているソフトウェアシステムである．

GUIの入力（操作）とシステム機能との関係はイベント駆動モデルで単純に表現でき，GUIの開発にも専用ツール[3]や仕様記述からの作成手段[4]などがあるが，GUIの出力制御はアプリケーションロジックの一部であり，一般化するのが困難である．また，GUI出力制御は一般的にイベント処理の中で行うが，1つのイベントの中で複数のGUI項目を制御することや，1つのGUI項目が複数のイベントの中で制御されることが必要である．

このためGUI出力処理はアプリケーションの仕様変更に弱く，また経験の少ないプログラマによる開発ではソースプログラムの可読性や悪く，バグの発生可能性が高くなる．

アプリケーションの仕様変更に対する柔軟性を図るため，GUI部とアプリケーションロジック部の分離方式[5]が提案されているが，アプリケーションロジック

†1 会津大学先端技術研究センタ
Core and Information Technology Center, University of Aizu
†2 会津大学コンピュータ理工学研究科
Graduate School of Computer Science and Engineering, University of Aizu
†3 会津大学コンピュータ理工学部
Department of Computer Software, University of Aizu
†4 山形大学工学部情報科学科
Department of Informatics Faculty of Engineering, Yamagata University

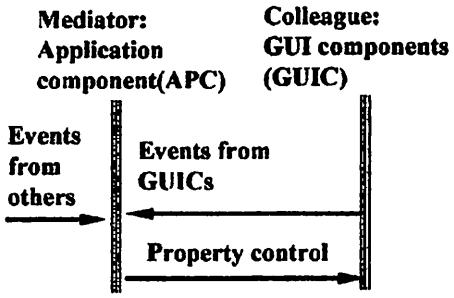

図 1 イベント駆動 GUI 制御モデル
Fig. 1 Event driven GUI control model

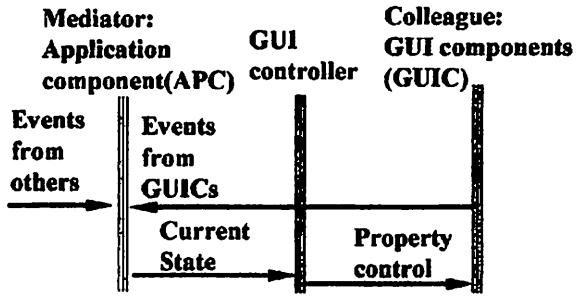

図 2 イベント・状態駆動 GUI 制御モデル
Fig. 2 Event-condition driven GUI control model

中の GUI 制御部分の仕様変更に対する考慮はなされていない.

本論文は GUI 出力制御の仕様変更に対する柔軟性を向上するための「イベント・状態駆動 GUI 制御機構」を提案する. この手法では, GUI 出力制御の条件となる**状態**を定義し, ユースケースで規定された制御内容から, イベントと状態との関係および状態と制御対象 GUI 項目の関係を抽出し, さらに状態に従った GUI 項目の制御を共通機能として実現する. これによりアプリケーションの中では個々のイベントの中での GUI 出力制御処理は制御条件の元となる状態の抽出に単純化することができる.

提案する手法は筆者の大学で開発されている遠隔教育支援システム RIDEE[6] の設計に応用され, システム仕様変更に対する柔軟性やシステム開発の効率向上における有効性が確認された.

以下, 第 2 章ではイベント・状態駆動 GUI 制御機構について述べ, 第 3 章では Java による実装と RIDEE システムの設計での応用を実例で述べる. 第 4 章では考察を行う.

2. イベント・状態駆動 GUI 制御機構

ここでは従来のイベント駆動型 GUI 出力制御方式に触れ, イベント・状態駆動 GUI 制御モデルについて述べる.

2.1 従来のイベント駆動 GUI 制御

従来のイベント駆動 GUI 制御では, アプリケーションコンポーネントとその配下の GUI 項目との関係は**図 1** のように書くことが出来る.

アプリケーションコンポーネント配下の GUI 項目は, それらの制御に関わる処理が実行されるまでに存在しなければならない. またアプリケーションコンポーネントが受けとるイベントには, ユーザの GUI 操作で引き起こされるものと, それ以外のもの (例えば他のコンポーネントやプロセスからのメッセージ) がある. これらのイベントを受ける後の GUI 出力制御は以下のように抽象化することができる.

制御条件の判断 GUI 出力制御の条件が成立するか否かを判断する. 一般的に 1 つの GUI 属性の制御にはこれまでに受信した複数のイベントの影響を考慮する必要がある. この条件判断を実現する処理プログラムは一般的に過去のイベントの処理結果を参照する複数の条件判断文から構成される.

GUI 属性制御の実施 制御条件判断の結果に基づき, GUI 項目の属性値を変更する. 一般的に 1 つの条件が成立した場合複数の GUI 項目属性を同時に変更する.

2.2 イベント・状態駆動 GUI 制御

一方, イベント駆動モデルのアプリケーションコンポーネントの振る舞いは有限個数の**状態**の間で遷移するものと考えることができる. また個々の GUI 制御対象 (GUI 項目の属性) の値をこの状態の関数として定義することができる.

状態は複数の**状態変数**から構成され, またそれぞれの状態変数は二つ以上の**状態値**を持つ.

状態値はアプリケーションコンポーネントのイベント処理により更新される. 即ちイベントの発生は状態の遷移を引き起こし, さらに GUI 項目の属性変更処理にタイミングを与える.

このため, まずアプリケーションコンポーネントの状態を状態変数および各状態変数の状態値で定義し, さらに各状態における制御対象 GUI 項目属性の取るべき値を GUI 出力制御を行うための**制御情報**として抽象化すれば, GUI 項目属性の制御は共通ロジックとしてアプリケーションロジックから分離することができる. 即ち, 図 1 の GUI 出力制御のモデルは**図 2** に書き換えることができる.

アプリケーションコンポーネントはまず上記の制御情報と各状態変数の初期状態値 (即ちコンポーネントの初期状態) を GUI 制御コンポーネントに渡しておき, さらにイベントを受けたたびに, そのイベントに基づき状態値を調整し新しい状態値を GUI 制御コン

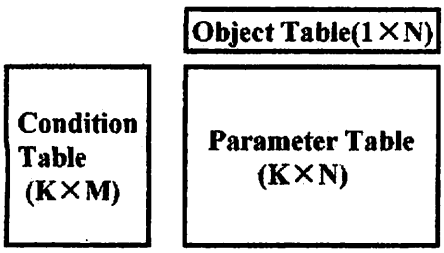

図 3 GUI 制御用情報
Fig. 3 Information for GUI control

ポーネントに通知する.

一方 GUI 制御コンポーネントはアプリケーションコンポーネントから各状態変数の状態値を受け取り, この状態値と制御情報から状態を特定し, それに基づき各 GUI 項目の属性を変更する.

2.3 GUI 制御情報

GUI 制御コンポーネントには, GUI 出力制御を遂行するために図 3 に示す制御情報が必要である.

2.3.1 Condition Table

Condition Table は制御のベースとなる状態を表す K 行 M 列の配列である. M は状態変数の数であり, K は状態の数である. Condition Table 中の個々の行 $C_i(i=1,K)$ は, 各々の状態での状態変数値を示す.

状態には状態変数中の一部にのみ依存するものがある. 即ちそれらの状態では一部分の状態変数のみが有効であり, そのほかの状態変数は状態の成立と無関係である. これを表現するために Condition Table の要素に**無効**という値を定義する. 即ち $c_{ij}(i=1,K;j=1,M)$ の値が無効であることは, i 番目の状態で j 番目の状態変数の状態値を問わないことを意味する.

Condition Table はアプリケーションコンポーネントから渡される各状態変数の状態値からそれの状態を特定するために利用される. どんな状態値の組み合わせでも一義的に状態を特定するためには, Condition Table の各行は**ユニーク**である必要がある. 即ちどの二つの行 C_a と C_b の間でもその各要素 c_{aj} と $c_{bj}(j=1,M)$ は以下のような関係にあってはいけない.

- $c_{aj}=c_{bj}$
- c_{aj} または c_{bj} が無効である

2.3.2 Object Table

Object Table は 1 行 N 列の配列であり, それの各要素は以下の制御対象情報を持つ.

- 制御対象 GUI 項目のオブジェクトへの参照
- 制御対象 GUI 項目属性の情報

同一 GUI 項目の異なる属性は Object Table では別々の要素として扱う. 但し同一 GUI 項目の同一属性は Object Table に 2 回以上表れてはいけない.

2.3.3 Parameter Table

Parameter Table は K 行 N 列の配列である. この配列の要素 p_{ij} は, アプリケーションコンポーネントの状態が Condition Table 中の i 番目状態に合致したときに, Object Table の j 番目の要素にある GUI 項目属性の持つべき値である.

Parameter Table の要素間は次のような関係にあってはならない.

- $p_{1j}=p_{2j}=,.....=p_{Kj}\{j=1,N\}$. これは状態と無関係に Object Table 中の j 番目の GUI 項目属性が常に同一であることを意味する.
- $p_{aj}=p_{bj}\{j=1,N\}$ になるような 2 つの行 P_a と P_b が存在する. これは Condition Table 中の 2 つの異なる状態で GUI が同一であること, 即ち無駄な状態が存在することを意味する.

3. 実装と応用

以下には Java によるイベント・状態駆動制御機構の実装および実際のシステム開発での応用について述べる.

3.1 GUI 制御コンポーネントの設計

制御コンポーネントは Java で guiController クラスとして実装した. コンストラクタを含め, guiController は以下のメソッドを提供する.

3.1.1 コンストラクタ

```
public guiController(
    String[][]  ct, // Condition Table
    guiObject[] ot, // Object Table
    String[][]  pt  // Parameter Table
);
```

コンストラクタで GUI 制御用の制御情報を guiController に登録する.

可読性と汎用性を図るために Condition Table の各状態値は文字列で定義する. 但し, 無効という状態値は空文字列で表現する.

Object Table の要素は guiObject 型のオブジェクトであり, guiObject は以下のように定義される.

```
public class guiObject{
    JComponent object,   // GUI object
    String  propertyCode // Property
);
```

但し guiObject の propertyCode は, 以下の何れの値であり, 制御対象の GUI の属性名を定義する.

visible GUI 項目の表示可否の設定
enabled GUI 項目の操作可否の設定

text　GUI 項目上に表示される文字列の設定

foreground　GUI 項目の文字色の設定

background　GUI 項目の背景色の設定

Parameter Table の各要素（GUI 項目の属性値）も同様文字列で定義する．guiController は Object Table 中の制御情報に従いこれらの文字列を適宜に変換する．**表 1** は上記 propertyCode の値と GUI 項目の属性値との関係を示す．

3.1.2 GUI 制御の実施

```
public void guiControl(
    String[]  conditionValues
);
```

conditionValues にはアプリケーションコンポーネントが管理している各状態変数の状態値が格納されている．guiController はこれらの状態値を用いて Condition Table から状態を特定し，さらに Parameter Table の値を用いて Object Table の各 GUI 属性を制御する．

3.1.3 診断制御画面の表示

```
public void showDiagnosisController();
```

showDiagnosisController は図 4 に示す画面を表示し下記の機能を提供する．

- 「Check Tables」ボタンをクリックすると guiController のコンストラクタで渡された制御情報の整合性がチェックされ，その結果は別画面で表示される．
- 画面上の Condition Table 中の行をクリックすると，その行で表わされる状態での GUI 制御をテストすることができる．
- 画面上 Condition Table の下側の状態変数値を変更すると，個別の状態値の組み合わせで GUI 制御をテストすることができる．

3.2 応　　用

提案した方法は筆者の大学の遠隔教育支援システム RIDEE の設計と開発に利用されている．以下には RIDEE の Floor Control 機能を実装した Participant Assistant(PA)[7] 画面の GUI 制御を例にしてその応用手順について述べる．

表 1　GUI 項目の属性値
Table 1　The properety value of GUI items

No	propertyCode	属性値	説明
1	visible	true/false	
2	enabled	true/false	
3	text	(文字列そのもの)	
4	foreground background	rrggbb	R,G,B の値 (16 進数)

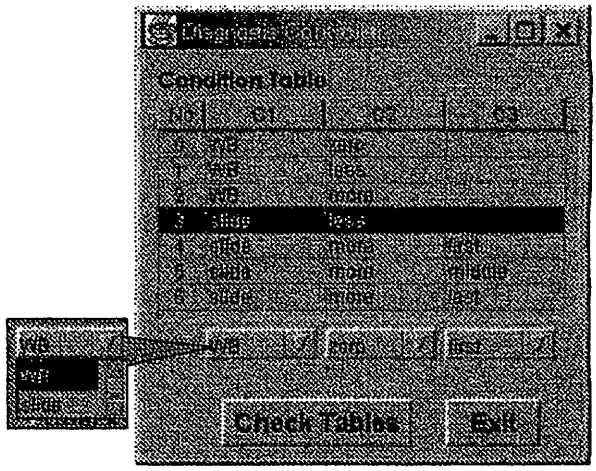

図 4　guiController の診断テスト支援画面
Fig. 4　The Diagnosis Controller of guiController

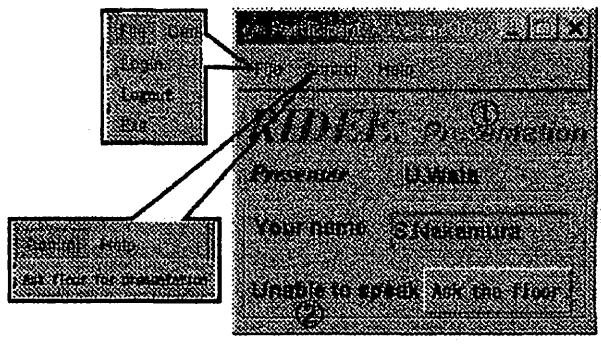

図 5　PA の画面
Fig. 5　GUI of the Participant Assistant

RIDEE は遠隔教育活動の参加者（教師と学生）全員が持つコンピュータで構成される分散協調処理システムである．これらのコンピュータの中では，1 台が司会者コンピュータであり，その他は参加者コンピュータである．PA は各参加者コンピュータに表示され，コンピュータの RIDEE への加入や，参加者がコンピュータを利用して発表または討論を行うときの発表権および質問権制御の支援を行う．図 3 は PA の画面を示す．

3.2.1 Object Table の構成

guiController の制御対象となる GUI 項目は以下の通りである．

RIDEE の状態　（図 3 の項目①）参加者に RIDEE 現在の状態を知らせる文字列．

Login ボタンおよび Logout ボタン　RIDEE システムへの Login および Logout に使用される．

Exit ボタン　自プロセスの終了に使用される．

Ask the floor for presentation ボタン　発表要求を司会者へ送る

発言可否の状態　（図 3 の項目②）PA の利用者に発言可能か否かを知らせる文字列．

Ask the floor ボタン　質問権を持っていない場合は，このボタンで質問要求を司会者へ送る．質問権を取得した場合は，このボタンで質問完了・取り下げ通知を司会者に送る．

上記 GUI 項目で構成される Object Table は**表 2** に示す．

3.2.2 Condition Table の構成

表 2 の GUI 項目の制御に影響を与える状態変数は以下の通りである．

接続状況　コンピュータが RIDEE に接続しているか否かを示す．「未接続」と「接続済み」の何れの値を持つ．

RIDEE の状態　司会者の制御下にある RIDEE システムの Working Mode[7] である．「Free」（司会者の制御なし），「OpenFloor」（発表者待ち），「Presentation」（発表中）および「Discussion」（討論中）の何れの値を持つ．

発表権の有無　この PA を持つコンピュータが発表を行うことがでるか否かを示す．「発表権取得済み」と「発表権無し」の何れの値を持つ．

質問権状況　質問者が存在するか否か，またはこのコンピュータが質問発言権を取得したか否かを示す．「質問者がいない」，「質問者がいるが自分でない」および「自分が質問権を持つ」の何れの値を持つ．

上記状態変数で構成される Condition Table **表 3** に示す．

3.2.3 Parameter Table の構成

表 2 と表 3 が決まれば，PA の外部仕様から Parameter Table の各要素を決めることができる．**表 4** は PA の Parameter Table 構成である．

3.2.4 PA のイベント処理

PA が受け付けるイベントには，以下のものが GUI 制御用状態変数の状態値に影響を与える．

接続状況変更　RIDEE の通信制御プラットフォーム[6] からのイベントであり，コンピュータの RIDEE への接続状況が変わったことを知らせる．

RIDEE 状態変更　司会者のコンピュータからの通知であり，RIDEE の Working Mode が変わったことを知らせる．

発表者情報　司会者コンピュータからの通知で，発表権を取得した参加者コンピュータの情報を通知する．

質問者情報　司会者のコンピュータからの通知であり，質問権を取得したまたは質問が完了・取り下げた参加者コンピュータの情報を通知する．

PA は以下のように，conditionValues に各状態変数の状態値を設定し，guiController の guiControl メソッドを呼び出す．

- 初期化処理では，conditionValues の 4 つの要素に {offline, free, no, none} を設定する．
- 接続状況変更イベントを受信したら，Login 成功の場合は conditionValues[0] に online を，それ以外の場合は conditionValues[0] に offline を設定する．
- RIDEE 状態変更の通知を受信したときは，通知中の RIDEE 最新状態に従い，conditionValues[1] に free, openFloor, presentation, または discussion の何れを設定する．
- 発表者情報の通知を受信したときは，通知中の情報で自分が発表者であるか否かを判断し，conditionValues[2] に yes または no を設定する．
- 質問者情報の通知を受信したときは，通知中の情報で質問者の有無と所在を判断し，conditionValues[3] に none, other または myself を設定する．

4. 考　　察

イベント・状態駆動 GUI 出力制御機構では，GUI 制御コンポーネントの導入により GUI 出力制御の処理をアプリケーションの処理から分離し，GUI 制御情報 (Condition Table, Object Table および Parameter Table) の抽象化により GUI 出力制御処理のロジックに関するプログラムを集中的に記述することができた．

表 2　PA の Object Table
Table 2　The object table of PA

No	制御対象 GUI 項目と属性	propertyCode
1	RIDEE 状態の文字列	text
2	Login ボタンの操作可否	enabled
3	Logout ボタンの操作可否	enabled
4	Exit ボタンの操作可否	enabled
5	Ask the floor for presentation ボタンの操作可否	enabled
6	発言可否状態の文字列	text
7	Ask the floor ボタンの操作可否	enabled
8	Ask the floor ボタン上の文字列	text

表 3　PA の Condition Table
Table 3　The condition table of PA

No	接続状況	RIDEE 状態	発表権有無	質問権状況
1	offline			
2	online	free		
3	online	openFloor		
4	online	presentation	yes	
5	online	presentation	no	
6	online	discussion	yes	
7	online	discussion	no	none
8	online	discussion	no	other
9	online	discussion	no	myself

表 4 PA の Parameter table
Table 4 The parameter table of PA

Condition No.	Object No.							
	1	2	3	4	5	6	7	8
1	offline	true	false	true	false		false	Ask the floor
2	online	false	true	false	true		false	Ask the floor
3	openFloor	false	false	false	true	Ready to ask floor for presentation	false	Ask the floor
4	presentation	false	false	false	false	Ready to speak	false	Ask the floor
5	presentation	false	false	false	false	Unable to speak	false	Ask the floor
6	discussion	false	false	false	false	Ready to speak	false	Ask the floor
7	discussion	false	false	false	false	Ready to	true	Ask the floor
8	discussion	false	false	false	false	Unable to	false	Ask the floor
9	discussion	false	false	false	false	Ready to speak	true	Stop speak

またこれらの制御情報は GUI の詳細設計段階で作成可能であり，制御情報の各要素が分かりやすい文字列表現で記述することができるため，設計者と実装者双方で理解でき，実際ではより精確な GUI 機能定義として仕様書の一部とすることができる．

Condition Table では 1 つの状態変数が複数のイベントの影響を受ける可能性があるが，1 つのイベントの処理中では複数の状態変数に亘るような処理が少ない．またアプリケーションコンポーネント内でのイベント処理は，イベント情報から状態値への変換だけとなる．このため，GUI 出力制御の仕様変更が発生した場合，大部分の対応作業は制御情報の変更に抑え，アプリケーションコンポーネント側のロジック変更を最小限にすることができる．

例えば状態変数や状態値の変更を伴わない仕様変更が生じた場合は，Parameter Table の修正で対応できる．また制御対象 GUI 項目に変更が生じた場合は，Object Table 中の要素とそれに対応する Parameter Table の列の修正で対応できる．さらに状態変数や状態値に変更が生じた場合は，Condition Table と Parameter Table の修正が必要でるが，アプリケーションコンポーネントでは，変更対象となる状態変数に関わるイベント処理のみの変更となる．

GUI 制御コンポーネントが提供した診断制御画面では制御情報の整合性チェックを行うことができるほか，状態値の個別変更を通して GUI 制御ロジックの単体テストを行うことができる．このため以下の問題は単体テスト段階で検出できる．

- 制御情報要素（文字列）中のスペリングエラー
- 未対応 GUI 項目の指定
- 未対応 GUI 属性の指定

GUI 制御コンポーネントは制御対象 GUI 項目同士間の参照が必要となる処理や，Parameter Table 内の要素（属性値）が動的変わるような処理を対応対象外としている．このような処理を対応するのに新たにプロトコルの定義が必要となり，かえって提案手法が利用しにくくなるからである．

5. おわりに

GUI 出力制御処理が簡単に実現できる手法として「イベント・状態駆動 GUI 制御機構」を提案した．またこの制御手法の有効性は筆者のソフトウェアシステム開発中で確認された．今後は本提案と従来方法の比較を行い，また GUI 制御自動設計支援機能（制御情報の可視化と専用エディタなど）や本モデルの GUI 制御それ以外への適用について検討する予定である．また現在の実装では状態数が多くなると状態特定のための検索負担が大きくなる．この問題の対策を講じる予定である．

参考文献

1) Java home page: http://developer.java.sun.com/
2) http://www.microsoft.com/japan/windows/default.asp
3) JBuilder home page: http://www.borland.co.jp/jbuilder/
4) 満田ほか：操作情報を中心としたユーザインタフェースの設計法，電子情報通信学会ソフトウェアサイエンス研究会報告 SS99-45(1999).
5) 鷲崎ほか：モデル／ビュー分離アーキテクチャ BeaM の機構とその評価，情報処理学会論文誌 Vol.41,No.10,pp.2454-2465 (Oct.2001).
6) Aiguo He ほか: リアルタイム遠隔教育環境の通信制御プラットフォームの設計，マルチメディア通信と分散処理ワークショップ論文集，Vol.2001, No.13, pp.231-218(2001).
7) A.He, et al: A Working Model for Real-time Interactive Distance Education Support Systems, IJCPOL, vol. 15 no. 2, June 2002.