プロトタイプのビジュアル・カスタマイズによるGUIの自動生成手法

白銀純子[†] 深澤良彰[†]

多種多様なアプリケーションが開発されるようになり，それに付随するグラフィカル・ユーザ・インタフェース（GUI）の構造も非常に複雑になっている．そのため，GUIアプリケーションの開発は，開発者にとって大きな負担となっている．本研究では，このような開発者の負担を軽減する1つの手法として，ダイアローグからのGUIアプリケーションの生成に対する新しいアプローチを提案する．本手法では，ダイアローグ記述の方法としてペトリネットを選択し，ペトリネットに最少限のGUIの情報を持たせてGUIアプリケーションの自動生成を行う．生成されたGUIアプリケーションの外観については，既存のツールを用いてユーザの好みにあわせることができることが大きな特徴である．

Method of Automatic GUI Generation with Visual Customization for Prototypes

Junko Shirogane[†] and Yoshiaki Fukazawa[†]

The demand for Graphical User Interface (GUI) applications has witnessed an increase. As the structure of GUI has become complicated more and more, it becomes difficult to develop such GUI applications. We propose an approach for the generation of GUI applications in order to facilitate GUI applications' development using dialog description. These dialogs are described using extended Petri net which includes minimum GUI information. The most important feature of our approach is as follows: after generating GUI application, the detailed size and arrangement of widgets can be customized to suit the user's requirements using existing tools.

1. はじめに

近年，豊かなグラフィカル・ユーザ・インタフェース（GUI）を持ったソフトウェアが多く求められている．しかし，GUIアプリケーションの構造は非常に複雑であるため，開発者にとって大きな負担となっている．このGUIアプリケーション開発において，我々は，次の2点が特に重要であると考えている．

条件1 アプリケーション全体の中で，各処理局面に応じて，必要とされる構成要素（ウィジェットと呼ぶ）を持ったウィンドウを配置すること

条件2 人間工学に基づいた機能性や操作性を考慮し，ユーザの要求や好みを反映した（以降，ユーザビリティと呼ぶ）GUIが与えられること

従来のアプリケーション開発では，ウィンドウを1つ1つ個別に作成し，それを組み合わせることによってアプリケーションを開発するため，ウィンドウ間の関連やウィンドウを越えたウィジェット間の関連は考慮することが困難であり，結果として用途の同じウィジェットを複数作成したり，必要なウィジェットを欠いてしまうなど，アプリケーションの正常な動作に支障をきたすことがある．したがって，各ウィンドウに適切なウィジェットを配置するための支援が必要となる．また，アプリケーションの外観は，ユーザが感じるアプリケーションの使いやすさと密接にかかわっており，どのような外観であれば使いやすいかということは，個々のユーザの好みや経験などに大きく依存する．これまでのGUI生成系によって自動生成されるウィジェットの配置・形状は，あらかじめ定められたアルゴリズムにより配置・形状の決定を行っており，ユーザの特殊性を反映することができない．本研究では，この2点を満たすソフトウェアを開発するための1つの手法として，GUIアプリケーションの生成に対する新しいアプローチを試みる．

これまでのGUIアプリケーション開発支援の分野

† 早稲田大学理工学部情報学科
Department of Information and Computer Science, Waseda University

においては，ダイアローグ記述，視覚的言語，GUI アプリケーションの自動生成，対話的な GUI アプリケーション構築ツール，コンポーネントによるアプリケーション構築ツールなどが研究されている．

ダイアローグ記述に関する研究[1]では，ユーザイベントを処理してユーザとアプリケーションとの対話を制御することを主な目的としており，状態遷移図や文法を用いるものなどが提案されている．しかし，効率的に GUI アプリケーションを開発するための支援については大きな注意が払われていない．

視覚的言語では，Prograph[2]や VisaVis[3]などが研究・実用化されている．これらの言語は，テキストベースの言語と同様に，アプリケーションの全体的な制御構造を把握するための支援がなされていないため，前述の条件 1 および 2 を満たすアプリケーションを開発することは困難である．

GUI アプリケーションの自動生成に関する研究では，データベースから自動生成するもの[4]やプログラムから自動生成するもの[5]などが研究されている．しかしこれらは，ウィンドウの動的な振舞いの制御のための支援を欠いているなど，実用性に欠ける面がある．またこれらは，アプリケーションの全体的な制御構造を把握するための支援を行っておらず，前述の条件 1 および 2 を満たすアプリケーションを開発するためには不十分である．

対話的に GUI を構築するツールとしては，Visual Basic[6]や Delphi[7]などのイベント駆動型アプリケーション構築ツールなどがある．これらのツールは GUI アプリケーションのイベント駆動型以外のソフトウェアや，GUI 部分以外のアプリケーション部分の開発については支援していない．また，アプリケーションの制御構造全体を把握するための支援を行っていないため，アプリケーション内の各処理局面において，ウィンドウに必要なウィジェットを過不足なく配置されているかどうかを容易に確認することは困難である．

コンポーネントによるアプリケーション構築ツールとしては，Java Studio[8]，APPGALLERY[9]などが製品化されている．しかしこれらは，あくまでコンポーネントという標準化されたプログラムを組み合わせてアプリケーションを作成するものなので，標準化が困難な業務固有のアプリケーションでは利用できない．

以上をふまえて，本論文では，次の 2 点を中心とする開発支援システムについての提案を行う．

- 条件 1 を満たすために，アプリケーション全体のダイアローグ記述から，必要な要素がすべて包含された GUI と，アプリケーションの制御の枠組みを生成する．
- 条件 2 を満たすために，本システムが生成した GUI を，ユーザの要求を満たすように，既存のツールを使って容易にユーザの要求を反映できる．

本論文では，2 章において本研究の概要，3 章で本システムのアーキテクチャを述べる．4 章から 6 章にかけて本システムを用いた詳細な GUI アプリケーション生成の流れを述べ，最後に評価および考察を行う．また本論文では，GUI アプリケーションの例としてビデオデッキの制御アプリケーションを取り上げ，この例を用いて説明を行っていく．

2. 本研究の概要

本研究では，与えられたダイアローグ記述を入力として GUI アプリケーションの骨組みを自動生成する．アプリケーションの使いやすさに大きくかかわるアプリケーションの外観については，個人の好みや経験などに大きく依存するため，外観についての情報を詳細に与えることはせず，自動生成後に，インタフェースビルダなどの GUI の外観設定をすることのできる既存のツールを用いてユーザの要求を反映する．

ダイアローグ記述の方法としては，定義が単純なペトリネット[10]を選んだ．これは，設計結果からのソフトウェア生成と同様に，GUI の自動生成の可能性の追求を主目的にし，これに対して実用性をそこなわないようなアーキテクチャ，入力方式を模索してきた結果，GUI アプリケーションの構造を記述するためにはペトリネットが適しているという結論に至ったためである．特に，多くの GUI アプリケーションでは，ユーザが複数の動作を選択的に起動し，それらを並行して操作できるように構築されており，ペトリネットでは，このような並行動作を容易に記述することができる．また，GUI アプリケーションは，ユーザがウィンドウに対して入力を行うと，それに応じて何らかの処理が行われるというイベント駆動型のものが多く，ペトリネットではトランジションを用いて，そのようなアプリケーションの挙動を表現することができるため，GUI アプリケーションの構造を記述するために適している．

以下において本研究の特徴について述べる．

2.1 外観変更

本システムでは，設計時に記述したペトリネットをもとに，仮の GUI を自動生成する．こうすることにより設計時と実装時の GUI の構成が一致することが保証される．しかしこの際の GUI の外観は，本システムが自動的に決定するため，ユーザビリティは十分

には考慮されていない．そこで，GUIアプリケーションの外観については，GUIが自動生成された後に，開発者がユーザと協議をし，ユーザビリティを考慮して変更を行う．この外観変更には特別なツールを必要とはせず，XF[11]やTclbombur[12]などの既存のツールを用いて行うことができる．この利点は以下のとおりである．

- アプリケーション設計時に必要なものは，アプリケーションの制御構造と，各制御の局面におけるウィンドウの構成情報であり，細かなウィンドウの外観の設定情報は不要であるべきである．
- 個々のウィンドウの外観について細かく，簡単に設定できるツールはすでに多く存在するため，ウィンドウの中に必要な構成要素が含まれていれば，外観設定は容易にできる．

2.2 アプリケーション開発における役割分担

GUIアプリケーション開発に携わる人を，

- ダイアローグ記述担当者
- GUI定義担当者
- プログラミング担当者

の3つに役割分担することを想定している．ダイアローグ記述担当者は，アプリケーション設計に関する十分な知識を持っており，アプリケーションのダイアローグ部の決定を担当する．GUI定義担当者は，ユーザビリティ，たとえば誤操作を減らすためにボタンやメニューを大きくする，あるいはボタンを小さくしてワークスペースを大きくする，などについて考慮し，GUIアプリケーションの外観を既存のツールを用いて変更することが業務である．プログラミング担当者は，アプリケーション本体の実装を行う．以上より，最初にダイアローグ記述担当者がダイアローグを記述すると，GUI定義担当者とプログラミング担当者は，それぞれ独立して作業を行うことができる．また，これらを担当する人の資質は互いに異なっており，明確な役割分担は，アプリケーションの効率的な開発において重要である．

2.3 ペトリネットの記法

通常のペトリネットにはウィンドウの構成などのGUIに関する情報が書かれておらず，また，GUIアプリケーションの制御構造を簡潔に記述するには不十分である．そのため本研究では，

- GUIアプリケーションの各処理局面で必要なウィジェットを定義し，そのウィジェットの最少限の情報をペトリネットに付加する．
- GUIアプリケーション開発に適したペトリネットの記法を新たに定義し，それを用いる．
- ペトリネットを記述するための様々な機能を持った専用エディタを構築し，それを用いてペトリネットを記述する．

2.4 ステートマネージャの生成

既存の手法では，設計結果を実装に反映させる際には手作業でマッピングを行わなければならない．この場合，設計結果と実装結果が一致しないという結果が起こりうる．そこで本研究では，自動生成されたGUIアプリケーションを，記述されたペトリネットに忠実に従って動作させるために，ステートマネージャと呼ぶ機構を用意した．ステートマネージャには，トランジションに対応づけられたすべての関数が登録されており，ステートマネージャはアプリケーション実行中のトークンの位置を監視し，発火条件の整ったトランジションの関数を起動する．ステートマネージャをアプリケーションに埋め込むことにより，自動生成されたGUIアプリケーションは，ペトリネットに忠実に従って動作することが保証され，設計結果と実装結果が一致することとなる．これによりプログラミング担当者は，トランジションに対応づけられている関数に関しては，その処理内容のみを記述すればよく，関数間の呼び出しなどの関係には注意を払う必要がなくなる．

以上より，GUIアプリケーションの開発に適した環境のもとでダイアローグ記述を行うことができる．

3. アーキテクチャ

本システムの全体的な構成を**図1**に示す．本システムの構成では，まず，新たに開発したペトリネットエディタ“PENGUIN”（Petrinet Editor for Navigator and Graphical User INterface）を使って，ダイアローグ記述担当者がペトリネットを記述する．この結果を保存すると，PENGUINは，プレース，トランジション，アークの座標を記述したファイル（物理ペトリネットファイルと呼ぶ）と，そのプレース，トランジション，アークの接続情報をテキスト形式で表現したファイル（論理ペトリネットファイルと呼ぶ）を作成する．このうち，論理ペトリネットファイルをPENGUINに用意されているGUI生成系にかけると，

(1) 各関数の骨格（処理内容の書き込まれていない関数とそのつながり）
(2) GUIプロトタイプ（ウィジェットの生成，ウィジェットに対するイベントの生起などのGUIに関するプログラム）
(3) ステートマネージャ

の3種類のプログラムが生成される．プログラミング

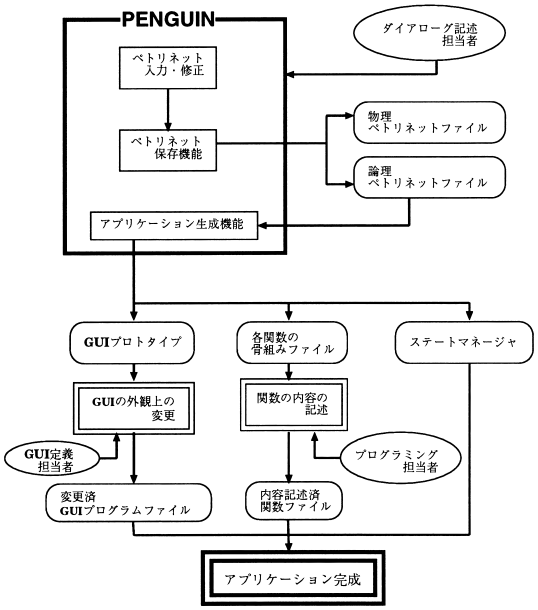


図 1 全体的な構成図
Fig. 1 System architecture.

担当者が実際の関数の中身（GUI 部分以外の処理内容）を (1) のプログラムの中に記述し，GUI 定義担当者が (2) の GUI プロトタイプを変更する．ステートマネージャには開発者側の変更は必要なく，この 3 者を結合することによりアプリケーションが完成する．

4. ペトリネット記述法

4.1 PENGUIN でのペトリネットの構成要素

本システムで使用するペトリネットは，一般的によく使用されるペトリネットであるプレース・トランジションネットに，抑止アークを加えた拡張ペトリネット[13]と呼ばれるペトリネットに，GUI アプリケーションに頻繁に使用される，または記述に必要となるペトリネットの表記法を追加定義したものである．抑止アークとは，このアークの入力プレースがトークンを持つ場合には，その出力トランジションを発火させないというアークである．**図 2** に PENGUIN でのプレース，トランジション，アークおよび抑止アークの表記法を示す．

4.2 新たに定義したペトリネットの構成要素

ボタントランジション ボタントランジションとは，ボタンが押されることによって発火するというトランジションである．GUI アプリケーションでは，表示されたウィンドウ上のボタンが押されることよって，関数が呼び出されたりウィンドウが表示されたりするなど何らか処理が行われることが非常に多い．

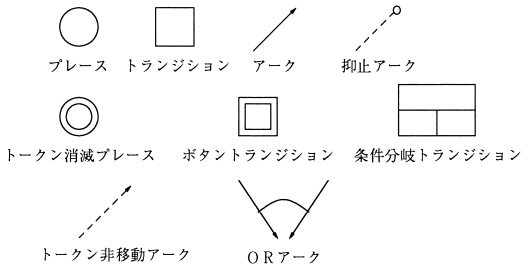


図 2 ペトリネットの表記法
Fig. 2 Notation of Petri net.

条件分岐トランジション 条件分岐トランジションとは，アプリケーション内のある処理の事前条件あるいは事後条件などによって，このトランジションの発火後のトークンを移動させるプレースが決まるトランジションである．たとえばビデオデッキのタイマの処理は，タイマボタンが押されたときに，タイマが “ON” の状態であれば “OFF” にし，タイマが “OFF” の状態であれば “ON” にする．トランジション自身のラベルを上段に，分岐条件のラベルを下段に書き込む．

トークン非移動アーク トークン非移動アークとは，アークの出力トランジションが発火すると，そのトランジションの出力プレースと，このアークの入力プレースの双方にトークンを移動させるアークである．たとえばビデオデッキの制御アプリケーションでは，操作ウィンドウ上において，再生ボタンを押してビデオを再生中に，タイマ設定ボタンを押してタイマ予約の設定を行うことができる．

OR アーク OR アークとは，あるトランジションの異なる 2 つ入力プレースとトランジションの間のアークを結び，その 2 つのプレースのうち，どちらか一方にトークンがあれば出力トランジションを発火させるというアークである．これを標準記法で記述すると，ペトリネットの図が複雑になってしまうため，このアークを定義した．

トークン消滅プレース トークン消滅プレースとは，このプレースに複数のトークンが配置されると，そのうちの 1 つのみを残してその他のトークンを消すというプレースである．

4.3 プレース（トランジション）の詳細情報

図 3 に，本システムのプレースの詳細情報入力ウィンドウを示す．トランジションの詳細情報入力ウィンドウも同様のものである．

本システムにおいて，プレースとトランジションが持つ情報を以下に示す．

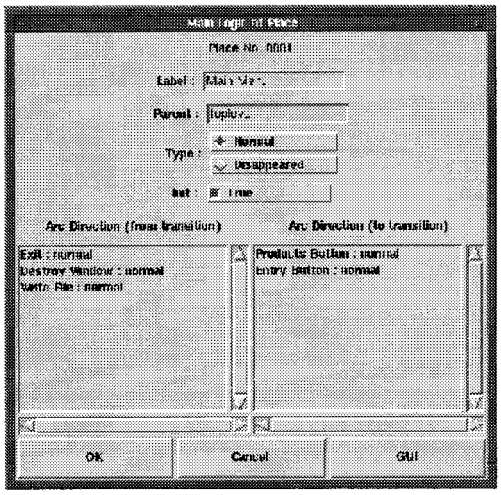


図 3 詳細情報設定ウィンドウ
Fig. 3 Window for place attribute.

プレース・トランジション共通

- それぞれを識別するための名前
- 階層構造になっている場合の親の名前
- 種類
- GUI 情報の有無
- 入力側の名前とアークの種類のリスト
- 出力側の名前とアークの種類のリスト

プレースのみ

- GUI アプリケーション起動時に，プレースが保持するトークンの数

トランジションのみ

- トランジションで実行される関数名（トランジションのみ）
- 分岐の条件の部分の名前（条件分岐トランジションのみ）

4.4 ペトリネットへの GUI 情報の付加

図 3 のウィンドウにおいて，“GUI” ボタンを押すとそれぞれプレース，トランジションの GUI 情報設定ウィンドウが表示され，GUI 情報を入力することができる．

4.4.1 プレースの GUI 情報設定

図 4 に，本システムのプレースの GUI 情報設定ウィンドウを示す．

プレースには GUI 情報としてウィジェット情報を持たせる．1 つのプレースに，アプリケーション内の 1 つのウィンドウを対応させ，そのウィンドウに配置するウィジェットを，ダイアローグ記述担当者が指定する．指定できるウィジェットとしては，ボタン，エントリ，メニューボタンなど 12 種類を用意している．

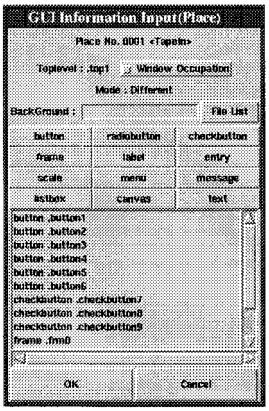


図 4 GUI 情報設定ウィンドウ（プレース）
Fig. 4 Window for widget generation.

指定されたウィジェットの一覧は，このウィンドウ中央にリストとなって表示される．ダイアローグ記述担当者は，そのプレースに必要なウィジェットを指定し，ウィジェットの名前やそのウィジェットでイベントが起こった場合の返値など，そのウィジェットに関する最少限の情報を入力する．このダイアローグ記述の段階では，ウィジェットの大きさやウィンドウ上での位置，色などの細かな外観に関する設定は行わない．

4.4.2 トランジションの GUI 情報設定

トランジションには GUI 情報として，ユーザイベントのバインド情報を持たせる．ダイアローグ記述担当者は，そのトランジションで起こるユーザイベントの種類（右ボタンクリック，フォーカス入力など）と，そのイベントをバインドするウィジェットを指定する．GUI 情報を持ったトランジションは，そのトランジションで指定されたイベントが起こることによって発火する．

ペトリネットでは，トークンによりその動作が制御される．したがって，アプリケーション起動時に複数のプレースにトークンを持たせたり，トランジション発火後に複数のプレースにトークンを移動させることにより，複数のウィンドウを同時に起動させることが可能である．また，本手法では，GUI 情報を持ったプレースがトークンを保持している場合に，ユーザはそのプレースのウィンドウに対して操作が可能となる．したがって，複数のプレースが同時にトークンを保持することにより，ユーザは複数のウィンドウを並行して操作することができる．

図 5 に PENGUIN を使って実際に入力したペトリネットの入力例を示す．この例は，ビデオデッキの制御アプリケーションで，ビデオの再生と録画予約の際の処理の流れの一部を表している．

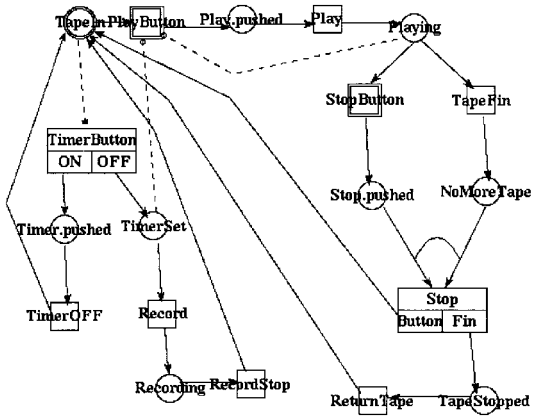


図 5 ペトリネット記述例
Fig. 5 Example of Petri net.

4.5 PENGUIN の補助機能

PENGUIN には，ペトリネットの記述を容易にしたり，記述の労力を軽減したりするためのいくつかの機能が用意されている．

4.5.1 ペトリネット記述時の機能

(1) ウィジェットの状態変化指定

この機能は，GUI アプリケーションの実行中にウィジェットの色や名前などの属性値などを変化させる場合に，どのように変化させるかを指定することができる機能である．たとえばビデオデッキの制御アプリケーションでは，録画予約のタイマが “OFF” の状態のときにタイマボタンを押すと，再生ボタンが使用可という状態から使用不可という状態になり，タイマボタンの表示が “Timer” という状態から “TimerSet” という状態に変化することが考えられる．このようにウィジェットの状態を変化させることは，アプリケーション利用者が誤操作を行うことを防いだり，利用者にとって使いやすいアプリケーションを作成するために非常に有効であり，大部分の GUI アプリケーションで採用されている．

本研究では，ウィンドウもウィジェットの 1 つであると考えており，この機能を用いることにより，特定のウィンドウに対して占有権を付与したり，その占有権を解除することも可能である．

このウィジェットの状態変化を PENGUIN で表現するには，状態変化が起こった直後のプレースにおいて，状態変化が起こる前のプレースを指定し，続いて状態変化の種類を指定することが必要である．

(2) プレース・トランジションの自動生成指定

この機能は，プレースの出力トランジション，およびトランジションの出力プレースを半自動生成する機能である．ペトリネットでは，1 つのプレースが複数の出力トランジション，あるいは 1 つのトランジションが複数の出力プレースを持っている場合がある．その複数の出力を 1 つ 1 つ記述していくということは，ダイアローグ記述担当者にとって大きな負担である．

このような場合，プレースの出力トランジションを，その種類と個数，そしてそれらをつなぐアークの種類を指定することによって，出力トランジションを生成することができる．生成されたトランジションは，ラベルなどの詳細情報を持っていないため，生成後にダイアローグ記述担当者が詳細情報を指定する．同様にして，トランジションの出力プレースを生成することもできる．

また，GUI アプリケーションでは，ボタンが押されることによって処理が行われたりウィンドウが表示されたりするなど，何らかの変化が起こることが多い．このため，あるウィンドウ上にボタンウィジェットが配置された場合には，そのボタンウィジェットの数と同数のボタントランジションが，そのウィンドウについての情報を持つプレースの出力トランジションとして，ダイアローグ記述担当者の指定なしで自動生成される．

(3) プレースの類似指定

1 つのアプリケーション内の異なる複数のウィンドウに配置されているウィジェットの種類やその個数などの構成が類似している場合がある．この機能はこのような場合に便利なであり，類似した複数のウィンドウのうちの 1 つを作成しておくと，その他は作成済みのウィンドウを保有するプレース名を指定するだけで作成できる．そして，作成したウィンドウを必要に応じて修正し，求める構成のウィンドウを作成することができる．

(4) ウィジェット変更機能

本システムでは，各ウィンドウにどのウィジェットを配置するかはダイアローグ記述担当者が決定する．しかし，GUI アプリケーションのユーザビリティを考慮して最終的な外観を決定するのは GUI 定義担当者の役割であるべきである．たとえば項目選択を行う場合に，ラジオボタンを使うかリストボックスを使うかなどというウィジェットの選択も GUI 定義担当者が行うべきものであると考えている．ウィジェット変更機能とは，ペトリネットが記述された後に，GUI 定義担当者が，ダイアローグ記述担当者が指定したウィジェットの種類を変更するための機能である．この機能では，たとえばラジオボタンからはリストボックスあるいはプルダウンメニューなどというように，個々

のウィジェットから変更することのできるウィジェットを限定することにより，ダイアローグ記述担当者が記述したペトリネットの構造を壊さないウィジェットを変更することができる．

5. GUIプロトタイプ生成

論理ペトリネットファイルから，PENGUIN中のアプリケーション生成機能がアプリケーションの骨組みとGUIプロトタイプを生成していく手順を以下に示す．

（1） 初期マーキングされているプレースを探し，そのプレースのGUIに関するプログラムを生成する．
（2）（1）のプレースを起点とし，アークをたどってトランジションを探索していき，(3)〜(6) を繰り返す．
（3） 発見したトランジションの出力プレースがGUI情報を持つ場合は（4），GUI情報を持たない場合は（5）の処理を行う．
（4）（3）のトランジションで呼び出される関数に，GUI部分の処理を行う専用の関数（GUI専用関数と呼ぶ）を挿入する．その後，(6) の処理を行う．
（5）（3）のトランジションからさらにアークをたどり，発見したトランジションで呼び出される関数を（3）で呼び出される関数の中に挿入する．その後（6）の処理を行う．
（6）（3）のトランジションからアークをたどってトランジションを探索し，発見されていないトランジションが存在した場合には（3）の処理を，すべてのトランジションを発見し終わった場合には（7）の処理を行う．
（7） 最後にGUI専用関数を作成し，その中にウィジェットの生成やウィジェットに対するイベントの生起などのGUIに関するすべてのコードを記述し，GUIプロトタイプを生成する．

6. プロトタイプの変更

2.1節で述べたように，アプリケーションの外観についてはPENGUINが自動的に決定している．ユーザビリティを考慮したGUIを構築するために，自動生成後に，GUI定義担当者が既存のツールを用いて外観変更を行う．この外観変更の段階で設定するものは

- ウィンドウ内でのウィジェットの位置や大きさ
- ウィジェットの色や文字のフォント

などの外観に関する事柄である．また，4.5.1項（4）で述べたウィジェット変更機能を用いることにより，アプリケーションの制御構造にかかわらないようにウィジェットを変更することも可能である．ただし，

- 新しいウィジェットの追加
- 既存のウィジェットの削除

といった，アプリケーションの制御にかかわるような設定は想定していない．本研究では，ダイアローグを用いてアプリケーションの制御構造を記述するため，この時点ではウィンドウ中に必要なウィジェットは過不足なくすべて含まれているということができる．したがって，この外観変更の段階では，GUI定義担当者はそれらの表現形式，配置，大きさなどの外観に気を使えばよい．

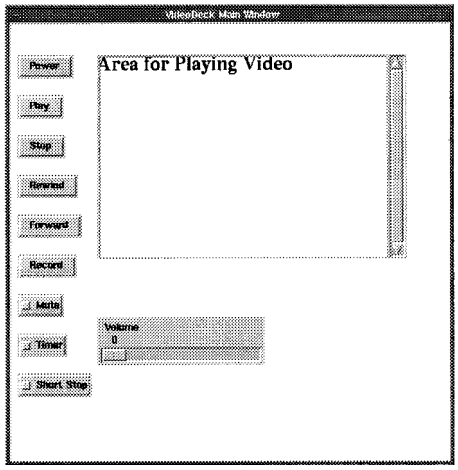
実際にPENGUINを使用して作成したプロトタイプのうち，XFを使用して変更したウィンドウの例を**図6**に示す．“Area for Playing Video” と表示されている領域でビデオの再生が行われる．（a）がPENGUINにより生成された変更前のウィンドウ，(b-1）および（b-2）が（a）を変更することにより作成したウィンドウである．(b-1）は，より楽しむためにビデオが映し出される場所を大きくとるという目的で，ボタンなどのビデオの操作にかかわるウィジェットを小さくしたものである．(b-2）は誤操作を減らすという目的で，ボタンなどのビデオの操作にかかわるウィジェットを大きくし，ウィジェット間の間隔をあけることにより，誤操作の減少を目指したものである．このようにウィンドウの外観変更のしかたはユーザの好みにより，複数存在する．

7. 評　　価

PENGUINを使用して，図5を含むビデオデッキのGUIアプリケーション，貸ビデオ店のGUIアプリケーション，および本システム “PENGUIN” という3つのアプリケーションを作成した．以下ではこの例を分析し，本手法の有効性を評価する．各例の規模を**表1**に示す．

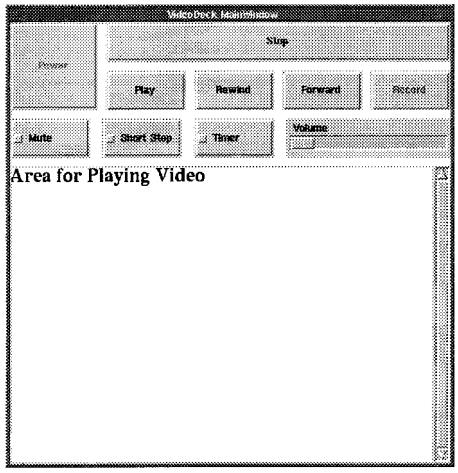
（1）開発したアプリケーションの性質

貸ビデオ店の管理アプリケーションを，PENGUINとプログラミング言語Tcl/Tkを用いる，Tcl/Tkによってすべてのプログラム・コードを記述する，VisualBasicを用いるという3つの手法で作成した．大部分の関数がユーザのイベントによって起動されるイベント駆動型のアプリケーションであり，かつ広く利用されている典型的なGUIアプリケーションであると考えて，貸ビデオ店のアプリケーションを取り上げた．作成しはじめてから完成するまでの時間と，生成したプログラムの総行数，アプリケーション内で宣言



(a)

(b-1)



(b-2)

図 6 外観変更例.(a) 変更前,(b-1),(b-2) 変更後
Fig. 6 Customized window.

表 1 ペトリネットと生成されたプログラムの内訳
Table 1 Petri nets and generated programs.

Application	ビデオデッキ	貸ビデオ	PENGUIN
プログラム（行）	476	712	2801
ウィジェット（個）	26	99	366
ペトリネット（枚）	6	11	32
全プレース	23	48	174
全トランジション	41	58	197
全アーク	76	133	483
GUI プレース	2	16	34
GUI トランジション	20	34	97
平均ウィジェット	13.50	5.50	10.76

表 2 各手法によるアプリケーション開発
Table 2 Comparison of development methods.

手法	PENGUIN +Tcl/Tk	Tcl/Tk	VisualBasic
時間（分）	460	520	710
行	3410	1018	2782
ウィジェット	168	170	133
関数	145	21	47

されている関数の総数を**表 2** に示す．PENGUIN で構築したアプリケーションの外観は XF を用いて変更した．このうち，PENGUIN+Tcl/Tk と Tcl/Tk の 2 つの手法によるアプリケーション構築では，プログラムの行数と関数の数に大きな違いが現れた．この理由は次の 2 点である．

- PENGUIN でプログラムを自動生成すると，GUI を表示するための専用の関数が生成される．この GUI 表示専用の関数の数は，初期マーキングのないプレースのうちの GUI 情報を持ったプレースと同数である．
- PENGUIN で自動生成したプログラムは，その GUI 部分が変更される．XF を用いて変更を行うと，1 ウィンドウの変更のために，ウィジェットの属性などの様々な情報が書き込まれた 2 つの関数が生成される．

しかし，増加している関数はすべて GUI の表示に関するものであり，アプリケーションの制御構造とは関係ないものである．したがってアプリケーションの制御構造が複雑化することはなく，増加した関数に関しては，プログラムの記述や保守の際には考慮する必要はない．アプリケーションの作成には，作成した人の各手法に対する熟練度などが評価結果に影響する．したがってこの結果は，本研究がアプリケーション開発にかかる時間の短縮に貢献するという証明にはならないが，他の手法と比べても実用的には問題ないものであることが分かる．

表3 ウィンドウ表示時間の比較
Table 3 Comparison of required time for displaying window.

手法	PENGUIN+Tcl/Tk	Tcl/Tk
ウィンドウ1	12165（9）	9863（9）
ウィンドウ2	36980（24）	33824（26）
ウィンドウ3	11726（17）	16717（16）
ウィンドウ4	8824（18）	7906（16）
ウィンドウ5	6281（11）	7304（15）

アプリケーションの実行時間に関しては，増加している関数やプログラムがGUI表示に関するもののみであるため，GUI表示のみに着目して実行時間の測定を行った．その結果が**表3**である．この測定は，PENGUINとTcl/Tkを用いる手法と，Tcl/Tkによってすべてのプログラム・コードを記述するという手法で作成した2つの貸ビデオ店のGUIアプリケーションを，Sun SPARC Station上でそれぞれ5回実行し，各ウィンドウが表示されるためにかかった時間を測定し，平均したものである．括弧内はそのウィンドウに配置されているウィジェットの数であり，測定時間の単位はマイクロ秒である．ただし，表3における各ウィンドウの比較は，その同様の構成をしたウィンドウを用いて行っている．

この結果を平均すると，PENGUINを用いて構築したアプリケーションでは，1個のウィジェットを表示するために約995マイクロ秒，すべてのプログラム・コードを記述するという手法で構築したものでは，922マイクロ秒かかるということになる．以上の結果より，PENGUINを用いて生成したアプリケーションをXFを用いて外観変更を行った場合に，プログラム・コードをすべて人手で記述したものよりもわずかに実行速度が遅くなることが分かった．しかしこれはアプリケーション利用者が遅いと感じるほどのものではなく，実用には問題ないものである．また，このように，Tcl/Tkという処理速度の遅いプロトタイプ言語を用いて実装を行った場合でも，実用的に問題ない実行時間が得られるため，他の言語を用いて実装を行った場合は，表3で示した時間以下でウィンドウが表示できると考えられる．したがって，この結果より，本手法を用いて開発したGUIアプリケーションは，GUIの表示時間については，実用には問題ないことが分かる．

（2）アプリケーション開発の手間

表1より，ウィジェット数を平均すると，1ウィンドウあたり9〜10個ほどのウィジェットを持っていることとなる．これらの例の中からいくつかのウィンドウを選んでXFを使用してウィンドウの外観変更を行ったところ，**表4**の結果が得られた．以上より3つの例の平均を求めると，9個ほどのウィジェットを持っている1ウィンドウを変更するためには，ウィジェットをどのように配置するか，あるいは外観変更に使用するツールに対するGUI定義担当者の熟練度などにもよるが，約6〜7分で十分であるということが得られた．

表4 ウィンドウの変更における手間
Table 4 Time required to customize window.

	時間（分）	ウィジェット（個）
例1	6	10
例2	21	28
例3	20	25
例4	13	20
例5	12	17

表5 自動生成されたプレース・トランジションの個数
Table 5 Number of generated places and transitions.

Application	ビデオデッキ	貸ビデオ	PENGUIN
プレース	15	4	47
トランジション	26	48	115
ボタントランジション	18	46	84

表6 プレースの類似指定
Table 6 Creation of similar places.

Application	ビデオデッキ	貸ビデオ	PENGUIN
全ウィンドウ数	2	16	34
基本ウィンドウ数	0	3	5
作成ウィンドウ数	0	9	26

（3）補助機能の有効性

3つのアプリケーションのペトリネットを記述する際に使用したPENGUINの補助機能のうち，**表5**にプレース・トランジションの自動生成指定，**表6**にプレースの類似指定についてその使用頻度を示す．

これらの表および表1の全トランジション数より，

- それぞれのアプリケーションにおいて使用されているトランジションのうち，約64%が自動生成可能であった．
- 自動生成されるトランジションの約78%がボタントランジションであった．

ということが分かった．以上より，プレース・トランジションの自動生成機能を用いてペトリネットを記述すると，ペトリネット記述の労力は大幅に削減できる．

表6では，それぞれのアプリケーションが持っているウィンドウの数（全ウィンドウ数）と類似指定を行う際にもととなったウィンドウの数（基本ウィンドウ数），類似指定を行って作成されたウィンドウの数（作成ウィンドウ数）を示している．たとえば，貸ビ

デオ店のアプリケーション（全16ウィンドウ）においては，3種類のウィンドウをもとに9ウィンドウが作成されている．この表より，アプリケーション内の約66%のウィンドウは，ウィジェットの種類やその数といった構成によって，少数（約15%）の基本ウィンドウのどれかに類似しているということが分かる．したがって，ダイアローグ記述担当者が一から作成するウィンドウはアプリケーション内の少数のウィンドウであり，その他の多くのウィンドウは，類似指定を用いて生成し，それを修正することにより作成することができる．

（4）**本システムの記述容易性**

ペトリネットと同様にアプリケーションの論理構造を記述することができる記述法の1つに状態遷移図がある．状態遷移図で記述できるものはすべてペトリネットでも記述できる[13]ので，記述能力ではなく，記述の容易性という観点からの比較が必要となる．

GUIの発展により，1つのウィンドウ内に多数のウィジェットが含まれるようになってきている．そして，これらのウィジェットへの操作に対する順序関係の制約が緩和されてきている．本システムを用いた設計においては，GUIの生成，およびGUI部とアプリケーション処理部との間のインタフェースの生成という観点から，アプリケーションの制御にかかわるウィジェットごとの記述が必要となる．たとえば，ビデオデッキの制御アプリケーションにおけるタイマ録画においては，ビデオテープの挿入と時間・チャンネルなどの設定の双方を行うことが必要であるが，この間の順序関係は規定されていない．この場合に2つの操作を明示した設計を状態遷移図を用いて行おうとすると，2つの操作がなされる順序のすべての組合せに対して状態を記述することが必要である．一方，ペトリネットでは，トークンの発火則を用いることにより，操作の順序の組合せを気にすることなく，容易に記述を行うことができる．本研究では，このような記述が頻繁に現れるようなアプリケーションを対象としており，ペトリネットが持つ記述容易性を選択した．

（1）〜（4）より，既存のGUIアプリケーション構築ツールによりアプリケーションを開発する，またはプログラムを記述していくなどの手間を考えると，GUIアプリケーション構築の労力は軽減されていることが分かる．

8. 終わりに

本システムは，開発者にとって大きな負担となっているGUIアプリケーションの開発労力を様々な面で軽減する目的で開発された．PENGUINでは，GUI情報を持ったペトリネットを使用してアプリケーションの制御構造を記述し，それをもとにしてGUIアプリケーションの自動生成を行う．

このときには以下のような利点が得られる．

- 複雑なGUIアプリケーションの開発が容易で，全体像を容易に把握することができる．
- ウィンドウの外観に関して，ユーザビリティを考慮したGUIを容易に構築することができる．

また，今後は

- 機能の追加・削除やGUI部のウィジェットの追加・削除など，ペトリネットの表面・内部における変更が生じるような保守を行う場合に，支援を行うシステムを構築すること
- コンポーネントウェア技術の概念を採り入れ，再利用性を向上させること

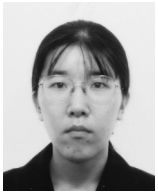
などについての拡張を行っていく予定である．

参考文献

1) Olsen, Jr., D.R.: User Interface Management Systems, Morgan Kaufmann Publishers (1992).
2) Cox, P.T., Giles, F.R. and Pietrzykowski, T.: Prograph: A Atep Towards Liberating Programming from Textual Conditioning, *IEEE Workshop on Visual Languages*, pp.150–156 (1989).
3) Poswig, J., Teves, K., Brankar, G. and Moraga, C.: VisaVis – Contributions to Practice and Theory of Highly Interactive Visual Languages, *IEEE Workshop on Visual Languages*, pp.155–161 (1992).
4) 白田由佳利，飯沢篤志：データベーススキーマ情報からのGUI自動生成，電子情報通信学会論文誌（D-I），Vol.J80-D-I, No.1, pp.71–80 (1997).
5) 北村操代，杉本　明：生成・カスタマイズ方式によるGUI構築手法の提案とクラスライブラリGhostHouseによる実現，情報処理学会論文誌，Vol.36, No.4, pp.944–957 (1995).
6) Microsoft: *Microsoft Visual Basic Programming System for Windows 95 and Windows NT*, Version 5.0, Microsoft (1997).
7) Borland International，アスキー書籍編集部（訳）：Borland公式コースウェアシリーズ・Delphi3オフィシャルコースウェアデベロッパー編，アスキー出版局 (1998).
8) JavaSoft: Java Studio. http://www.sun.com.
9) 長谷川裕行：APPGALLERYで超簡単 Windowsプログラミング入門，ソフトバンク (1997).
10) 青山幹雄，内平直志，平石邦彦：ペトリネットの理論と実践，朝倉書店 (1995).
11) Delmas, S.: XF. ftp://ftp.cs.tu-berlin.de.

12) SRA: Tclbombur. http://www.sra.co.jp.
13) 村田忠夫：ペトリネットの解析と応用，近代科学社 (1992).

(平成 11 年 6 月 28 日受付)
(平成 12 年 5 月 11 日採録)



白銀　純子

昭和 49 年生．平成 11 年早稲田大学大学院理工学研究科修士課程修了．現在，同大学大学院理工学研究科博士課程に在学中．GUI ソフトウェア開発支援の研究に従事．

深澤　良彰（正会員）

昭和 51 年早稲田大学理工学部電気工学科卒業．昭和 58 年同大学大学院博士課程中退．同年相模工業大学工学部情報工学科専任講師．昭和 62 年早稲田大学理工学部助教授．平成 4 年同教授．工学博士．ソフトウェア工学，コンピュータアーキテクチャ等の研究に従事．ソフトウェア科学会，IEEE，ACM 各会員．