6W-8 ユースケース記述にもとづくGUI開発手法

白銀 純子　深澤 良彰
早稲田大学理工学部

1 はじめに

ユーザの視点を取り入れてアプリケーションの設計・実装を行うことの重要性が高まってきている。中でもGUI（Graphical User Interface）に関しては、早期にプロトタイプを作成し、ユーザの意見を反映させることが欠かせなくなっている。ユーザの視点を反映させることに関しては、ユースケース図やシナリオが注目されている。ユースケース図やシナリオはユーザを中心として記述されるため、ユーザの視点を容易にアプリケーションの設計・実装に取り入れることができる。そこで本研究では、開発者がユースケース図及びシナリオを記述し、それもとにして、アプリケーションのGUI部のプログラムを生成する手法について提案する。

2 UMLとユースケース

本研究では、ユースケース図及びシナリオから、ウィンドウの構成要素（ウィジェット）の生成やウィジェットの属性の取得・設定のメソッド等を含むGUI部のプログラム（GUIプロトタイプと呼ぶ）を生成することを目指す。ユースケースとは、主にUMLを利用したアプリケーション開発の分析フェーズで利用されるモデル化手法である[1]。図1及び図2に、本研究で用いるユースケース図とシナリオの例を示す。これは、単純化した銀行のATMで預金をする際の例である。モデル化対象のシステムとやりとりをする人や物をアクター（図1:“預金者”）と呼ぶ。アクターとシステム間の対話をモデル化したものをユースケース（図1:“定期預金をする”及び“普通預金をする”）と呼ぶ。アクターとユースケースとの間の関係を図示したものをユースケース図（図1）と呼ぶ。さらに、ユースケースにおけるイベントの流れの1つを記述したものをシナリオと呼ぶ（図2）。また本研究では、現時点ではユースケースの包含・拡張に関しては考慮しておらず、アクターは人に限るものとする。

3 GUIプロトタイプ生成手順

GUIプロトタイプを生成するためのステップを図3に示す。本研究では、ユースケース記述(Step 1)、シナリオ併合グラフ生成(Step 2)、シナリオ併合グラフ修正(Step 3)、入力・表示項目指定(Step 4)、GUIプロトタイプ生成(Step 5)という5つのステップでGUIプロトタイプを生成する。以下において、この5つのステップについて詳しく述べる。

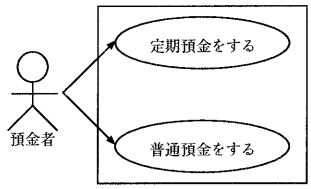

図1: ユースケース図例

ユースケース: 定期預金をする 1. 預金をするを選択する。 2. 定期預金を選択する。 3. 口座番号を入力する。 4. お金を入れる。 5. 金額を確認する。 6. 続けるを選択する。	ユースケース: 普通預金をする 1. 預金をするを選択する。 2. 普通預金を選択する。 3. 口座番号を入力する。 4. お金を入れる。 5. 金額を確認する。 6. 続けるを選択する。
ユースケース: 定期預金をする 1. 預金をするを選択する。 2. 定期預金を選択する。 3. 口座番号を入力する。 4. お金を入れる。 5. 金額を確認する。 6. やめるを選択する。	ユースケース: 普通預金をする 1. 預金をするを選択する。 2. 普通預金を選択する。 3. 口座番号を入力する。 4. お金を入れる。 5. 金額を確認する。 6. やめるを選択する。

図2: シナリオ例

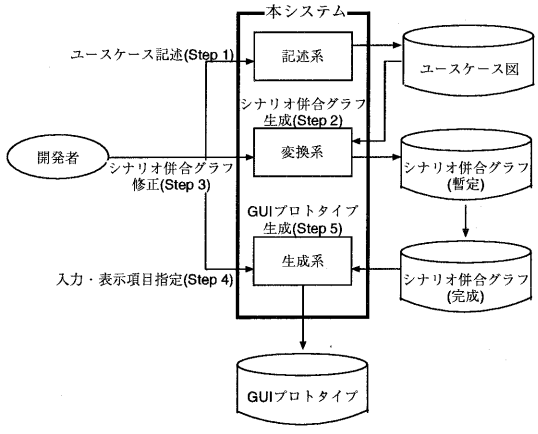

図3: GUIプロトタイプ生成

3.1 ユースケース記述

開発者が本システムを用いてユースケース図及びシナリオを記述し、それを本システムの変換系に入力する。シナリオは自然言語で箇条書きで入力することができる。この際に、必ずしも全てのイベントが異なる動作を表すわけではなく、同一の意味を表すものも多数存在する可

GUI Development Method from Use Case Description
Junko Shirogane and Yoshiaki Fukazawa.
School of Science & Engineering, Waseda University.

能性がある。そこで、各シナリオ内の各イベントについて、どのシナリオのどのイベントと同一の意味を表すかを指定することができる（同一イベントコピー機能）。こうすることにより、シナリオ記述の労力を軽減することができる。

3.2 シナリオ併合グラフ生成

GUIプロトタイプを生成するためには、入力されたユースケース図及びシナリオから、アプリケーション全体の中でのユーザイベントの流れを把握する必要がある。ここでは、入力されたユースケース図及びシナリオをもとにして、ユースケースの併合を行う。その後、併合されたユースケースのシナリオ内のイベントをノードとする有向グラフを生成する。このとき、開発者によってあらかじめ指定された、同一の意味を表すイベントは同一ノードとして統合しておく。また、開発者に指定されていなくても、全く同じ文字列で記述されているイベントは同一の意味を表すものとみなし、同一ノードとして統合する。詳細なアルゴリズムについては、4章において述べる。できあがった有向グラフをシナリオ併合グラフと呼ぶ。2章の銀行のATMの例において作成されたシナリオ併合グラフを図4に示す。

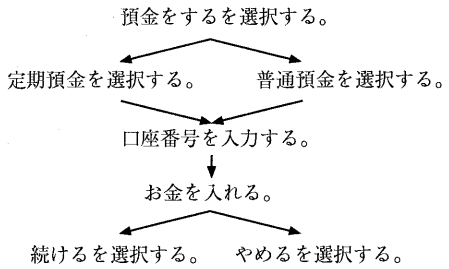

図 4: シナリオ併合グラフ例

3.3 シナリオ併合グラフ修正

開発者によって指定される、シナリオ内の同じ意味を表すイベントに関して、必ずしも同一イベントコピー機能を利用して指定されているとは限らない。そこで、できあがったシナリオ併合グラフを開発者に示し、意味的に同じ内容のイベントを指定し、同一化する、誤って指定してしまったイベントを訂正するなどの修正を行う。これにより、シナリオ併合グラフが完成する。

3.4 入力・表示項目指定

ユースケース図とシナリオのみでは、ユーザが入力する項目やユーザに表示する項目を本システムが把握することはできない。そこで開発者が、各シナリオ上の入力・表示項目と、どの入力・表示項目が同一ウィンドウ上にあるかを指定する。入力・表示項目に関しては、入力か表示かを指定する。入力項目の場合にはキーボードからの入力か選択枝による選択か、また、キーボードからの入力の場合は文字列か数かを指定する。

3.5 GUIプロトタイプ生成

シナリオ併合グラフ修正ステップで完成したシナリオ併合グラフ及び入力・表示項目指定ステップで指定された入力項目及びウィンドウについての情報をもとにGUIプロトタイプを生成する。各入力・表示項目のために利用するウィジェットは、入力・表示項目指定ステップで指定された入力・表示項目の種類をもとに本システムが自動的に決定する。

4 シナリオ併合グラフ生成アルゴリズム

図3の Step 2 における、ユースケースの分割・統合を行うためのアルゴリズムは以下の通りである。

```
for(i=0;i<ユースケース数;i++){
  /* ユースケースの併合 */
  ユースケース内のユースケースの第1イベントを読む;
  if(これまでに読んだイベントと同じイベントがある)
    その同じイベントを持つユースケースに、
    このユースケース内のシナリオを統合する;
}
for(i=0;i<併合後のユースケース数;i++){
 /* シナリオ併合グラフ生成 */
 for(j=0;j<シナリオ数;j++)
  for(k=0;k<シナリオ内のイベントの数;k++){
   k+1番目のイベントを読む;
   if(k+1番目のイベントは別のイベントと同じ意味を
     表すという情報が開発者から与えられている)
    k+1番目に行われるイベントが、指定されている
    別のイベントであるという情報を格納する;
   else if(k+1番目と同じイベントが存在する)
    k+1番目に行われるイベントが
    発見されたイベントであるという情報を格納する;
   else
    k+1番目に次に行われるイベントが
    k+1番目のイベントであるという情報を格納する;
 }
```

5 終りに

本研究では、ユーザの視点を取り入れてGUIの設計・実装を行うことを目的として、ユースケース図及びシナリオからGUIのプロトタイプの自動生成を行う手法について提案した。今後の課題としては、

- 同じ機能を表現しながら、シナリオの第1イベントの記述が違うユースケースの扱い
- 開発者の意図とは異なるGUIプロトタイプが生成された場合の対処
- ユースケースの包含・拡張記述への対応

などが挙げられる。

参考文献

[1] テニー・クアトロニー著、日本ラショナルソフトウェア株式会社監訳: “UML 活用型開発入門 -Rational Rose を用いた実践的ケーススタディ -”、株式会社トッパン