ユーザの操作順序におけるユーザビリティ評価のためのプロトタイプ生成

柴田博成[†1]　白銀純子[†2]　岩田一[†3]　深澤良彰[†1]

アプリケーション内でユーザが行う操作の順序は，システムの学習容易性，効率性，記憶容易性など，ユーザビリティと密接な関係がある．本研究ではユーザの行う操作の順序に着目し，ユーザビリティの高いアプリケーションを開発することを目標として，ユーザが行う操作の順序を基にGUIプロトタイプを自動生成する手法について提案する．これにより，開発者がアプリケーションのユーザビリティを確保する労力の削減を目指す．

Prototype Generation for Usability Evaluation of User Event Flows

HIROSHIGE SHIBATA[†1]　JUNKO SHIROGANE[†2]
HAJIME IWATA[†3]　YOSHIAKI FUKAZAWA[†1]

The event sequences of user interaction affect usability of an application, such as learnability, efficiency and memorability. In this paper, focusing on operation sequences, we propose a method of generating GUI prototypes using event sequences of scenario for developing an application with high. Due to our method, it is possible to reduce costs of developing application with high usability.

1. はじめに

アプリケーションのユーザビリティについて，使用しているウィジェットの種類，レイアウト，操作の一貫性，ユーザの行う操作の順序などが重要視されている事項として挙げられる[1]．特に，アプリケーション内でユーザの行う操作の順序は作業の効率性や学習容易性に大きく関わり，ユーザビリティの良し悪しに大きく影響する．

ユーザの行う操作の順序は，アプリケーション開発において，ユースケース図とシナリオで表現することが多い．シナリオはユーザがアプリケーションを操作するときの順序を忠実に表現している．操作の順序がユーザにとって自然な流れでないと，そのアプリケーションは使いにくい製品となる．ゆえにシナリオを自然な順序で定義する事はユーザビリティの面において重要である．しかし，自然な操作の順序とは業務内容やユーザによって大きく異なり，順序を定義するのは困難である．

自然な操作の順序を定義するには，実際に完成したアプリケーションあるいは試作段階のアプリケーション，つまりGUIプロトタイプを直接ユーザが使用し，問題があればGUIプロトタイプを構築，改良しながらイベントを修正する事が効果的である．しかし，ユーザが実際に使用できるシステムは開発者の労力やコストの面から，ユーザビリティの評価を十分に行えるほど早期に構築することは非常に困難である．

そこで本研究では，ユーザの行う操作の順序に着目し，シナリオに基づいたアプリケーションのGUIプロトタイプの構築を支援する手法を提案する．生成されたGUIプロトタイプを使用し，操作の順序についてGUIプロトタイプを利用したユーザビリティ評価を行うことを目標とする．具体的には，自然言語で記述されたシナリオを本研究で設計するシステムへの入力として，シナリオのイベントの順序に沿ったGUIプロトタイプを自動生成し，早期にユーザが利用できるGUIプロトタイプの構築を可能とする．そのGUIプロトタイプをユーザが利用し，操作の順序に問題がある場合や，イベントの欠如や無駄がある場合には，ユーザがシナリオ内のイベントの追加，削除，訂正，順序の変更を行う．ユーザテストのみで開発者がソースコードに直接触れることなく，容易にGUIプロトタイプの変更ができるため，開発者の労力とコストを削減できる．そして，早期のGUIプロトタイプ生成を行うことにより，操作の順序におけるユーザビリティ評価を支援することを目標とする．

2. シナリオ

2.1 シナリオとは

ユースケースを実現するためのフローの各々をシナリオと呼ぶ．各シナリオはアクタと呼ばれる，モデル化対象のアプリケーションとやりとりをする人や物の視点で記述される[2]．本手法では，シナリオ内の1つ1つのステップをイベントと呼ぶ．ユースケースについてのイベントフローの中で最も典型的なフローであり，終始うまく進むフローで記述したものを主シナリオと呼ぶ[3]．言い換えれば，バグやエラーが発生せずに事後条件を満たすシナリオである．図1に，主シナリオの例を示す．この例は，インターネットバンキングのシステムで「振り込みをする」の主シナリオの例である．

†1　早稲田大学
Waseda University
†2　東京女子大学
Tokyo Woman's Christian University
†3　神奈川工科大学
Kanagawa Institute of Technology University

ユースケース：振り込みをする	
1	システムは顧客に振込元の口座番号と暗証番号の入力を求める
2	顧客は振込元の口座番号を入力する
3	顧客は暗証番号を入力する
4	システムは顧客に振り込め詐欺が急増している旨の注意喚起をする
5	システムは顧客に振込先の情報の入力を求める
6	顧客は振込先の金融機関を選択する
7	顧客は振込科目を選択する
8	顧客は振込先の口座番号を入力する
9	システムは顧客に振込先情報の詳細を示す
10	システムは顧客に振込依頼人名の入力を求める
11	顧客は振り込みの依頼人名を入力する
12	システムは顧客に振り込み上限金額が 50 万円である旨を示す
13	顧客は振り込み金額を指定する
14	システムは顧客に最終確認を表示する
15	顧客は完了を選択する

図 1 「ユースケース：振り込みをする」の主シナリオ例

1 つのユースケースにおけるフローは 1 つとは限らず，主シナリオ以外に複数のシナリオが存在する場合がある．複数の処理の流れをそれぞれシナリオとして記述し，1 つのユースケースに複数のシナリオを持たせることが多々ある．主シナリオ以外のシナリオは副シナリオという．副シナリオの記述は基本的に主シナリオの記述と同じ作業である．副シナリオは，主シナリオについての代替およびエラーとなる内容を扱う．

2.2 本手法におけるシナリオの定義

本研究では，自然言語で記述されたシナリオを基に，ユーザの行う操作の順序に沿った GUI プロトタイプを自動生成する．提案する手法では，自然言語で記述したシナリオを本システムへの入力とし GUI プロトタイプを自動生成するために，本手法で用いるシナリオの記述について定義する．図 1 は以下の定義に従って記述したシナリオである．定義の内容を以下に示す．

- 1 イベントは 1 文とし，述語は 1 つとする．
- ユーザが入力する項目やユーザに出力する項目（これらを総称して以後インタラクション項目と呼ぶ）は 1 イベントにつき 1 項目とする．
- 必要なインタラクション項目の粒度は最小単位で扱う．
- 排他的選択や複数選択の際の各選択項目はシナリオ内に記述されず，別途，各イベントに対する付加情報として記述する．
- 副シナリオは，主シナリオと独立したシナリオとする．

3. 本研究の特徴

開発工程の早期に行えるユーザビリティ評価

ユーザテストは，基本的で欠かすことのできないユーザビリティの評価手法である．ユーザテストを行うことによって，人々がどのようにコンピュータを使用するか，インタフェースの問題点などについて具体的な情報を得ることができる[1]．また開発者が実際に，ユーザの課題解決場面を観察する事ができるため，ユーザビリティの問題点を発見しやすいほか，開発者とユーザの見解の違いが解消できる．

本手法では，要求分析段階においてシナリオを用いる事で，イベントの順序に沿った GUI プロトタイプの生成を行う．これにより，開発の早期に GUI プロトタイプを使ったユーザテストによるユーザビリティ評価を行える．さらに設計・運用段階において，ユーザ側からの GUI に対する修正要望の削減につながることも期待できる．

GUI プロトタイプ生成にかかるコストの削減

開発者側のコストの面から，ユーザビリティの評価を十分に行えるほど早期にユーザが実際に利用できる GUI プロトタイプの構築は困難である．さらに，ユーザテストから得られたフィードバック結果を GUI プロトタイプに反映，修正する必要がある．

本手法では，GUI プロトタイプを生成する上で，自然言語で記述されたシナリオを利用し，各々のイベントに対して要求仕様を設定するのみで GUI プロトタイプを生成する．また，操作の順序についてユーザテストを行った結果をシナリオに反映させることで，GUI プロトタイプの改良が可能となる．開発者が直接ソースコードに触れることなく，シナリオの訂正のみで GUI プロトタイプの生成を行えるため，GUI プロトタイプの構築にかかる開発者の労力やコストを抑えることが期待できる．

ユーザの操作順序に沿った GUI プロトタイプの自動生成

アプリケーション内でユーザの行う操作の順序は，ユーザビリティを考慮する上で重要な要素である．特に，ユーザが行う作業の効率やシステムの使用方法の習得しやすさに大きく関わり，ユーザが感じるアプリケーションのユーザビリティの良し悪しに大きく影響する．ユーザの行う操作の順序は，一般にシナリオ内でイベントの順序として記述されるが，シナリオの記述段階では，記述されたイベントの順序でユーザビリティ上の問題が無いかを判断する事は困難である．

本手法では，GUI プロトタイプがイベントの順序に沿って生成されるため，実際に GUI プロトタイプを利用した上でユーザの行う操作の順序におけるユーザビリティ上の問題について考察する事ができる．GUI プロトタイプの改良を繰り返し行うことで，イベントの順序とウィジェットについてのユーザビリティ向上につながる．

4. 本研究の詳細

本手法は，要求分析の成果物であるシナリオから GUI プロトタイプを自動生成し，ユーザが行う操作の順序に基づくユーザビリティ評価を行うことを目指す．本手法を実現するためのシステムの構成図を図 2 に示す

本手法は以下の流れで GUI プロトタイプの生成及び改良を行う．

(1) シナリオの入力
(2) GUI 情報の設定
(3) GUI プロトタイプの生成
(4) ユーザビリティ評価
(5) ユーザビリティ評価結果の反映

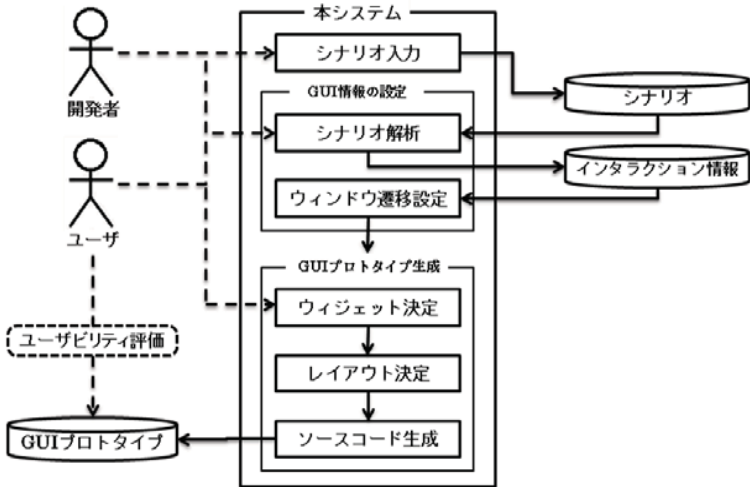

図 2 本システムの構成

4.1 シナリオの入力

本研究ではまず，自然言語で記述されたシナリオを本システムに対して入力する．本手法の定義に従って記述されたシナリオを 1 イベントずつ順番に入力する．排他的選択や複数選択の際の各選択項目などの情報はシナリオ内に記述せず，別途，各イベントに対する付加情報としてシステムへ入力する．

4.2 GUI 情報の設定

4.1 節の段階では，本システムへの入力が自然言語で記述されたシナリオのみであり，GUI プロトタイプを生成するために必要な情報を得る事が出来ない．そこで，本システムへシナリオを入力した後，イベントごとにインタラクション項目やイベント形式，ウィジェットを構成する情報，ウィンドウが遷移するタイミングなどを設定する．設定は，ユーザもしくは開発者が行う．

4.2.1 インタラクション項目とイベント形式の設定

イベント内には，ユーザがシステムへ入力する項目，もしくはシステムからユーザへ出力される項目が存在する．これらの項目を総称してインタラクション項目と呼ぶ．例えば，「顧客は振込科目を選択する」というイベントは，「振込科目」がインタラクション項目である．インタラクション項目は各イベントの文字列の中に記述されているため，文字列の中からインタラクション項目をユーザもしくは開発者が抽出する．

イベントの入出力の実現方法は個々で異なる．この入出力の実現方法をイベント形式と呼ぶ．例えば，図 1 のイベント 8 :「顧客は振込先の口座番号を入力する」は，「ユーザからの入力イベントであり，キーボードから文字列を入力する」というイベント形式である．入力フィールドのサイズや排他的選択や複数選択の選択項目などの GUI のウィジェットを決定する際に必要な補足情報もイベント形式として設定する．

以上のように，イベントごとに，文字列の中からインタラクション項目を抽出し，その項目がどのようなイベント形式になるかを，ユーザもしくは開発者が設定する．

インタラクション項目とイベント形式の関係の一部を表 1 に示す．図 1 のイベント 7 について，インタラクション項目とイベント形式の設定例を図 3 に示す．

表 1 インタラクション項目とイベント形式の関係

インタラクション項目	イベント形式		
入力イベント	入出力方法		補足設定
	キーボードから入力	文字列	入力フィールドのサイズ
	選択	排他的選択	各選択の項目の設定
		複数選択可	各選択の項目の設定
		数値（排他的選択）	上限値，下限値，数値の間隔，単位の設定
出力イベント	表示	文字列表示	表示する文字列の設定

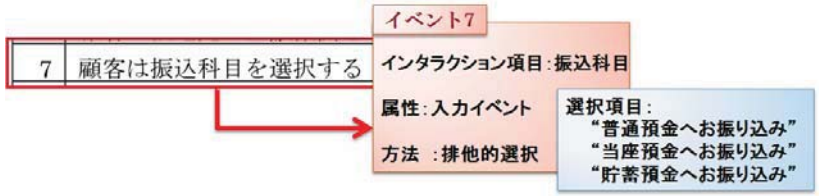

図 3 インタラクション項目とイベント形式の設定例

4.2.2 各ウィンドウに配置するイベントの設定

シナリオ内の全てのイベントついて，インタラクション項目，イベント形式の設定後，どのイベント直後に次のウィンドウへ遷移させるかをユーザもしくは開発者がイベント単位で設定する．例えば，イベント 12 の「システムは顧客に振り込み上限金額が 50 万円である旨を示す」と，イベント 13 の「顧客は振り込み金額を指定する」という 2 つのイベントでは「振り込み上限金額」，「振り込み金額」がインタラクション項目である．この 2 つのインタラクション項目が同じウィンドウに属する場合，この 2 つのイベントが同一ウィンドウを構成するという設定をする．

4.3 GUI プロトタイプ生成

各イベントに対して設定されたインタラクション項目とイベント形式の情報を基に，本システムが GUI プロトタイプを生成する．各ウィンドウに配置するウィジェットの種類とレイアウトは，本システムが自動的に決定する．

4.3.1 使用するウィジェットの決定

各イベントに対して決定したインタラクション項目とイベント形式の情報を基に，本システムが GUI プロトタイプ内で使用するウィジェットを決定する．

Microsoft Windows のユーザインタフェースのガイドライン[4]に基づき，ウィジェットの適用基準を設ける．イベント形式に対し適用可能なウィジェットが複数存在する場合は定義した適用基準に従い，使用するウィジェットをシステムが自動で判別する．

例えば，図 1 のイベント 7 について，図 3 のようにインタラクション項目とイベント形式を設定した場合，入力方法が排他的選択であるため，対応するウィジェットはラジオボタンかコンボボックスとなる．イベント 7 では，排他的選択で選択項目数が 3 個である．ガイドライン[4]に基づき，排他的選択項目数が 8 個未満の場合ラジオボタン，8 個以上の場合コンボボックスを使用するため，ラジオボタンが適用される．

4.3.2 ウィンドウの遷移

設定されたウィンドウ分割の情報に基づき，4.3.1 で決定したウィジェットを各ウィンドウに配置する．そして，ウィンドウの遷移を実現させるために「戻る」と「次へ」の 2 つのボタンを各ウィンドウに配置し，遷移の処理として結びつける．

4.3.3 ソースコード生成

図 1 のシナリオに対して，ユーザもしくは開発者によるインタラクション項目とイベント形式とウィンドウ分割の設定，本システムによる適用するウィジェットとレイアウトの確定後，Java 言語による GUI プロトタイプのソースコードを自動生成する．

図 1 のシナリオについて，ウィンドウ 1 にイベント 1~3 を，ウィンドウ 2 にイベント 4 を設定した際のウィンドウ 1 とウィンドウ 2 のウィンドウ遷移の例を図 4 に示す．

図 4　GUI プロトタイプのウィンドウの遷移例

4.4 ユーザビリティ評価

本システムによって自動生成された GUI プロトタイプをユーザや開発者が実際に利用してユーザビリティ評価を行う．具体的には以下の項目について評価を行う．

- イベントの順序
- 不足しているイベントや無駄なイベントの発見
- ウィンドウが遷移するタイミングの適切性
- 利用するウィジェットの適切性

4.5 ユーザビリティ評価結果の反映

GUI プロトタイプをユーザが利用し，イベントの順序に問題がある場合や，イベントの欠如や無駄がある場合には，シナリオの修正を行う．具体的には，ユーザもしくは開発者がシナリオ内のイベントの追加，削除，訂正，順序の変更をするのみで，開発者がソースコードに直接触れることなく，容易に GUI プロトタイプを変更できる．

インタラクション項目とイベント形式を決定し，イベント 5〜8 を同一ウィンドウとして設定し，生成される GUI プロトタイプの例を図 5 の左側に示す．

GUI プロトタイプをユーザが利用してユーザビリティ評価を行った結果，例えば図 1 のシナリオについてイベントの順序を入れ替える必要がある場合を想定する．イベントごとのインタラクション項目とイベント形式の設定は保持されているため，ユーザはイベントの順序を入れ替えるのみで GUI プロトタイプの改良ができる．イベントの順序を変更に伴い，従来のウィンドウ分割を変更する必要がある場合には，改めてインタラクション項目ごとに配置するウィンドウを決める必要がある．

また，同じウィンドウ上に他のドロップダウンリストが存在する場合，一貫性を保つためにラジオボタンをドロップダウンリストに変更する事がユーザビリティに効果的な場合が多い．本システムでは開発者によるソースコードの修正を行わずにイベン

ト形式の設定に戻り，適用するウィジェットを予め設定しておくことを可能とする．

図 1 のシナリオのイベント 6 と 7 を入れ替え，イベント 7 のウィジェットをラジオボタンからコンボボックスに変更した際の GUI プロトタイプ例を図 5 の右側に示す．

図 5　イベントの順序とウィジェットを変更した GUI プロトタイプ

イベントの順序とウィジェットについて，改善案がある限り自動生成される GUI プロトタイプの改良を繰り返し行うことにより，ユーザビリティの向上が可能となる．

5. 評価

本手法の有用性を確認するため，2 つのシステムのシナリオから 1 つずつを用意し，本手法を適用した．用意したシナリオの 1 つは，図 1 のインターネットバンキングのシステムについての主シナリオ「振り込みをする」である．もう 1 つは TV 番組予約システム[5]の主シナリオ「録画を予約する」である．本手法に基づいて自動生成した GUI プロトタイプについて，以下の項目の確認を行った．

- 本手法を用いた場合と手動でコーディングした場合の，GUI プロトタイプ生成までに費やした時間の比較
- 本手法を用いた場合と手動でコーディングした場合の，GUI プロトタイプの修正（イベントの追加・削除・順序の変更）に費やした時間の比較
- 本来適用される事が予定されているウィジェットと本手法により自動生成した GUI プロトタイプに適用されているウィジェットとの比較

5.1 GUI プロトタイプ生成に費やす時間の比較

本研究は，開発者がソースコードに触れずに，容易に GUI プロトタイプを生成する手法を提案している．開発者の労力やコストを抑えられるかを検証するために，本研究で開発したシステムを使ってシナリオから GUI プロトタイプを自動で生成するまでに費やす時間と，手動でコーディングを行い GUI プロトタイプの生成をするまでに費やす時間を比較した．本手法と手動でコーディングは共に Java 言語を用いている．

本システムを利用して GUI プロトタイプを自動で生成する時間を計測する実験では，シナリオが完成しており，各イベントに対してインタラクション項目・イベント形式・ウィンドウ分割の要求が獲得できている状況で行った．そして，本システムの起動から GUI プロトタイプ生成までの時間を対象とした．

手動でコーディングを行い，GUI プロトタイプを生成する時間を計測する実験では，統合開発環境の Eclipse [6]を用いてコーディングを行った．ウィジェットの種類・サイズ・レイアウト，ウィンドウの分割方法が決定している状況下において GUI プロトタイプ生成までの時間を対象とした．

本手法を用いて自動で生成した場合と，手動で生成した結果を表 3 に示す．

表 3　GUI プロトタイプ生成に費やす時間の比較

	コンポーネント数*1	イベント数	手動生成	自動生成
「振り込みをする」	45 個	15	1 時間 47 分 18 秒	10 分 42 秒
「録画を予約する」	35 個	10	1 時間 22 分 56 秒	7 分 06 秒

手動でのコーディングに比べ，本手法を用いて GUI プロトタイプの生成する際は，約 10%の時間で行える結果であった．実験の結果，自動生成に費やす時間はシナリオのイベント数に比例すると予測できる．また，手動生成に費やす時間は，イベント数に依存せず，GUI プロトタイプを構成するコンポーネントの総数に比例すると予測できる．以上より，開発者側の労力やコストを抑え，容易に GUI プロトタイプを生成でき，本手法の有用性が高いことが証明されかつ，開発対象のアプリケーション内に GUI のコンポーネントが多いほど，本手法が有効であるといえる．

5.2 GUI プロトタイプ修正に費やす時間の比較

5.1 節で生成した「振り込みをする」GUI プロトタイプについて，イベントの追加・削除・順序の変更をした際の GUI プロトタイプ修正に費やす時間を，本手法を用いた場合と手動でコーディングの場合を比較した．表 4 は 1 イベントの追加・削除・順序の変更を毎回異なるイベントを対象として 10 回行い，平均値を算出した結果である．

追加・削除・順序の変更について，本手法にて修正を行う事で，手動でコーディングする修正の時間を約 8%～19%に抑えられる結果となった．実際の開発では，イベント数も膨大になり，ソースコードはさらに複雑となるため，本手法を用いた GUI プロトタイプの修正は今回の評価実験の結果以上の有効であると考えられる．

*1　GUI プロトタイプを構成するコンポーネントの総数

表 4　シナリオ修正に伴う GUI プロトタイプ修正に費やす時間

	追加	削除	順序変更 （ウィンドウの変更無し）	順序変更 （ウィンドウの変更有り）
手動生成	368 秒	59 秒	64 秒	317 秒
自動生成	42 秒	5 秒	12 秒	35 秒

5.3 GUI プロトタイプを構成するウィジェットの比較

本手法により自動生成された GUI プロトタイプで適用されているウィジェットが，実験前に想定していたウィジェットの種類と一致しているかを検証した．

「振り込みをする」のシナリオについては，本来適用を予定しているウィジェット 15 種類が全て一致していた．インターネットバンキングのシステムで適用されるウィジェットは，Windows のガイドライン[4]に沿って適用している．本システムはこのガイドラインに沿って GUI プロトタイプを生成しているため，期待した結果が得られた．

「録画を予約する」の GUI プロトタイプについては，実際のアプリケーション[5]のウィンドウを想定して比較した．このシナリオについては，排他的選択について使われるウィジェットの種類が異なった．排他的選択項目数が 10 項目のイベントである際，本手法で自動生成したウィジェットはリストであったが，自動生成されたウィジェットはコンボボックスである．TV 番組予約システムは TV の画面の大きさと，チャンネルがインタフェースであるため，Windows のガイドライン[4]が適していなかった．しかし，以上の結果より，インターネットバンキングのシステムでは全て，TV 番組予約システムでは排他的選択以外のウィジェットは当初の想定と一致していた．したがって，本手法でインタラクション項目に割り当てるウィジェットは，大部分が適切なものであるといえる．

6. 関連研究

小形ら[7]は，要求分析モデルからの GUI プロトタイプ自動生成により，要求分析モデルと GUI プロトタイプの整合性を保証する手法を提案している．小形氏ら[7]が提案している手法は，まず，顧客の要求と既存の業務を文章と図で導出する．これをユースケース図に書き起こし，ユースケース単位で定義したシステム化後のアクティビティ図，入出力のグループ化としてクラス図，インスタンス仕様の定義としてオブジェクト図を連携させることにより，要求分析モデルを設計している．この要求分析モデルから GUI プロトタイプを生成し，さらに入出力項目の表示に具体的なデータを表示させ，直観的な理解の容易さを向上させる手法を述べている．

Vi ら[8]は，タスクモデル，ドメインモデル，ユーザーモデルの 3 つ叙述的モデルを利用し GUI プロトタイプを生成する事で，UI を低コストで効果的に評価できる手法を提案している．

小形ら[7]と Vi ら[8]の手法は，GUI プロトタイプを生成するまでに開発者が要求分析モデルを確実に定義する必要があり，時間と手間など，開発者の負担が大きい．また，顧客からの GUI プロトタイプの変更要求があった場合には，まず要求分析モデルの修正が必要であり，即座の対応が難しい．本研究の手法では，シナリオのみを用いてユーザが容易に GUI プロトタイプの生成と改良に携わることができるため，開発者にかかる負担を軽減できるだけでなく，ユーザからの的確な要求が獲得可能である．

7. おわりに

本研究では，ユーザの行う操作の順序がユーザビリティに大きく影響を与える点，操作の順序についてのユーザビリティ評価には GUI プロトタイプの生成および改良が必要な点，GUI プロトタイプの構築と改良にはコストがかかる点に着目した．シナリオから GUI プロトタイプの構築を自動化し，操作の順序について GUI プロトタイプを利用したユーザビリティ評価を行う手法について提案した．

今後の課題としては，本手法の GUI プロトタイプを生成について手動で行っている部分を自動化する手法の提案，複数のシナリオから 1 つの GUI プロトタイプの生成，GUI プロトタイプを使ったユーザビリティ評価方法の検証などが挙げられる．

参考文献

1) Jakob Nielsen，篠原稔和監訳，三好かおる訳：ユーザビリティ・エンジニアリング原論，東京電機大学出版局 (1999).

2) John M. Carroll，郷健太郎訳：シナリオに基づく設計，共立出版 (2003).

3) Geri Schneider, Jason Winters，羽生田栄一監訳，オージス総研訳：ユースケースの適用：実践ガイド，ピアソン・エデュケーション (2000).

4) Windows User Experience Interaction Guidelines
http://msdn.microsoft.com/ja-jp/library/aa511258.aspx

5) 一般社団法人 電波産業会 文字放送による TV 番組録画予約システムの規格
http://www.arib.or.jp/english/html/overview/doc/2-BTA_T-003v1_0.pdf

6) Eclipse
http://www.eclipse.org/

7) 小形真平，松浦佐江子：UML で記述された要求分析モデルからのプロトタイプ自動生成，第 6 回情報科学技術フォーラム, pp.107-110 (2007).

8) Vi Tran, Jean Vanderdonckt, Manuel Kolp : Generating User Interface from Task User and Domain Models, Proc. of Second International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services, pp.19-26 (2009).