ヒューマンインタフェース 92－9
（2001．1．18）

ユーザの操作履歴に基づくGUI適応化の試み
～XMLで記述したGUIグラフに基づく適応処理～

寺岡 照彦 京本 寿美恵† 中西 俊克† 押田 秀治† 秋吉 政徳

三菱電機株式会社 先端技術総合研究所 †同 系統変電・交通システム事業所

本稿では、GUI構成をExtensible Markup Language(XML)によってグラフ表現し、ユーザの操作履歴を基にして、適応化させる試みについて述べる。GUIの適応化は、グラフの一部を変更する、あるいは合成するといった更新作業によって実現する。適応化として、ユーザの一連の作業から、GUI構成全体をオフラインで再構成する手法と、ユーザが現在行っている作業を支援するための、オンラインでの予測インタフェース作成について説明する。オフライン再構成は、GP(Genetic Programming)をベースにした手法であり、特定の評価指標を最適にすりょうなGUI構成を求めるものである。オンライン予測インタフェースは、操作によって利用されたGUI要素を抽出し、その部分グラフを合成することによって作成するものである。

A study on adaptive user interfaces based on a user's interaction history

Teruhiko TERAOKA, Sumie KYOMOTO†, Toshikatsu NAKANISHI†,
Hideharu OSHIDA†, Masanori AKIYOSHI

Advanced Technology R&D Center, † Transmission & Distribution, Transportation Systems Center
MITSUBISHI ELECTRIC CORPORATION

This paper proposes the methods for adapting graphical user interfaces (GUIs) to each user based on the user's interaction history. A GUI is represented as a graph described by XML(Extensible Markup Language). The structure of the graph is modified using GP(Genetic Programming)-like method for reducing the total number of user's interactions with the software. The methods for dynamically updating a GUI are also presented. With the methods, new GUIs are generated by synthesizing GUI elements containd in the user's interaction history. The number of user's interactions can be reduced by these adaptive GUIs.

1. はじめに

最近のソフトウェアは高機能になった反面、必要以上に多機能化・巨大化し、ユーザにとっての使いやすさを必ずしも満たしているとは言えない。これに対して、ユーザやタスクに応じて、構成や動作を変更させる「適応型インタフェース」や「予測インタフェース」が注目を集めている[1, 2]。

ユーザにとって、長期的な視点からは、自分や仕事に応じたGUI構成が得られれば効率がよい。例えば、ソフトウェアの習熟によって必要のないGUIは出現しないようにしたり、良く使う機能だけでGUIが再構成されれば、効率がよくなるであろう。また、短期的な視点からは、特定の機能を繰返して利用する場合などに対して、自動マクロ作成など、操作の短縮が図れると効率がよい。前者に対しては、オフラインでGUI構成の再編を図る適応型のインタフェースが、後者に対してはオンラインで適応を図る予測型のインタフェースが向いていると考える。本稿では、GUI(Graphical User Interface)構成をXML(Extensible Markup Language)によってグラフ表現し、ユーザの操作履歴を基にグラフを更新することで、GUI適応化を図る試みについて述べる。

2. 適応型/予測インタフェース

適応型インタフェースとして、ワープロのかな漢字変換のようによく使うものを選択しやすくする機能や、ユーザの作業を監視して適切なヘルプを提供する機能が提案されている[3]。Microsoft社の

Office2000では、使わないメニュを見かけ上消すことで、メニュの選択効率をあげる適応を行っている。

予測インタフェースとしては、UNIXのシェルのように、コマンド操作の履歴を保存し、頭文字だけで再実行できる機能などが提供されている。PDAなどペンで文字入力する機器に対して、POBOX[4]は、簡単な相関と利用頻度によって、入力した文字列に続く文字列を、リアルタイムに予測して候補を与えるものである。単純な手法ながら、高速な文章入力が可能となる。

本研究では、ユーザの操作履歴を基にして、オフラインでのGUI再構成[5]と、オンラインでの予測GUI作成という2つのアプローチによって、ユーザ/タスクに適応するインタフェース作成について述べる。概要を図1に示す。

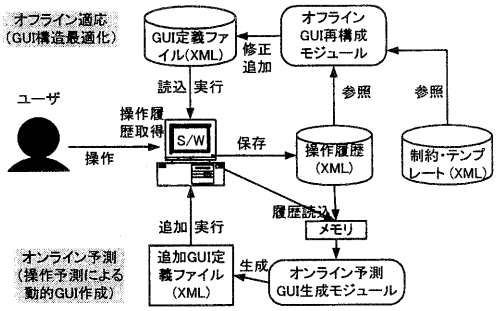

図1: 適応/予測インタフェース作成の概要図

3. GUIと操作履歴のXMLによる記述

本節では、XMLによって記述しているGUIおよび操作履歴について説明する。

3.1 XMLによるGUIの記述

ソフトウェアのGUI構造は、一般にグラフ構造で記述可能であるため、XMLによって記述した。XMLの利点として、下記の点があげられる。

- 変更や解析処理のために、DOM (Document Object Model) やSAX (Simple API for XML) などの、汎用のAPI/ツールを利用できる
- 階層構造をもったデータ (← GUI) を直感的に記述できる
- 将来的に、仕様書記述やGUI設計、プログラム設計を、統一的に扱える可能性がある

XML形式で記述されたGUIとアプリケーション機能要素を統合し、一つの実行可能プログラムを生成するライブラリとして、IBMのBML (Bean Markup Language) [6]や、Universal Interface Technologies のUIML (User Interface Markup Language)[7]などが開発されている。今回は、IBMのBMLを利用した。BMLによって記述したGUIメニュの例を図2に示す。

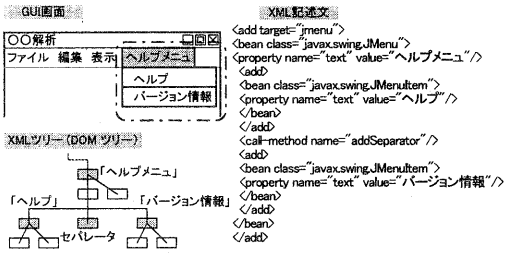

図2: XML(BML)によるGUIメニュの記述例

3.2 イベントの取得とXMLによる操作履歴記述

S/Wプログラムの操作履歴もXML形式で保存する。XML形式とすることで、タグによる処理が容易になる (例：あるGUI要素で発生したイベント情報の収集) が、他の形式で記述しても特に問題はない。例えば、下記のようなスタイルで操作履歴を保存する。

```
<event_list_set>: 操作系列の集合
 <event_list>: 操作系列
  <event>: 個別操作 (イベント)
   <jclass/>: GUI のクラス名
   <name/>: 名前/タイトル
   <value/>: 値 (on/off → 1/0)
   <device/>: 使用デバイス (マウス or キーボード)
   <text/>: キーボード入力した文字
   <container/>: イベントが発生した GUI 用を含むコンテナのタイトル
   <position x=''*'' y=''*''/> : マウス/キー入力時の座標値
   <date/>: イベントが発生した時刻
  </event>
  ...
  <event>...</event>
 </event_list>
</event_list_set>
```

本研究ではシミュレーションやプロトタイプ作成にJava/Swingを利用し、イベント取得にはJava Accessibility Utilities [8]を利用している。

4. オフラインでのGUIの適応化

グラフ表現された問題に対して、ある指標を (準) 最適にする解を求める手法として、GP (Genetirc

Programming) が知られている [9]。GP では木構造を扱い、木に対する変更オペレーションとして、(1) mutation: ノードのラベル変更、(2) inversion: 兄弟ノードの並べ替え、(3) crossover: 部分木の切り替え、の 3 つが適用される。これらのオペレーションを反復することで解を求めるものである。

従来から、GUI の構成変更は、設計された GUI の評価/テストを行って、コード修正するのが一般的である。本節では GP のオペレーションを基本にして、GUI グラフを変更し、オフライン適応する方法について説明する。

4.1 GUI グラフの変更オペレーション

GUI グラフの変更オペレーションとして、下記のものを考えた。これらのオペレーションを反復し、評価関数を最適化するような GUI グラフを求めることが、本研究のオフライン適応の主旨である。

1. 順序変更 (inversion): 配置/出現順序の変更。
   例) メニュ、コンボ・リストボックス、テーブルのカラム内の配置変更。ダイアログの出現順序の変更。
2. 交換 (crossover): 部分木の交換。
   例) 別ダイアログとのアイテム (GUI 要素) 交換。
3. クラス・属性変更 (mutation):
   例) プルダウン⇔ポップアップ。注目色の変更。
4. 移動 (move):
   例) メニュアイテムの他メニュへの移動。テキストフィールドの他ダイアログへの移動。
5. 統合 (merge):
   例) 2 つのボタン→ 1 つのボタン (コマンド統合)。2 つのダイアログ→ 1 つのダイアログ
6. 分割 (separation):
   例) ダイアログやプルダウンメニュの分割。
7. コピー (copy):
   例) メニュアイテム→ポップアップメニュやツールボックスのボタン。

図 3 は、上記のオペレーションをメニュを例として行うことを説明した図である。

4.2 制約条件とテンプレート

意味のなさない変更を抑制するためには、下記のような制約条件を設定すればよい。

1. グループ： GUI 要素のまとまり (ex. 機能別、作業別) の設定

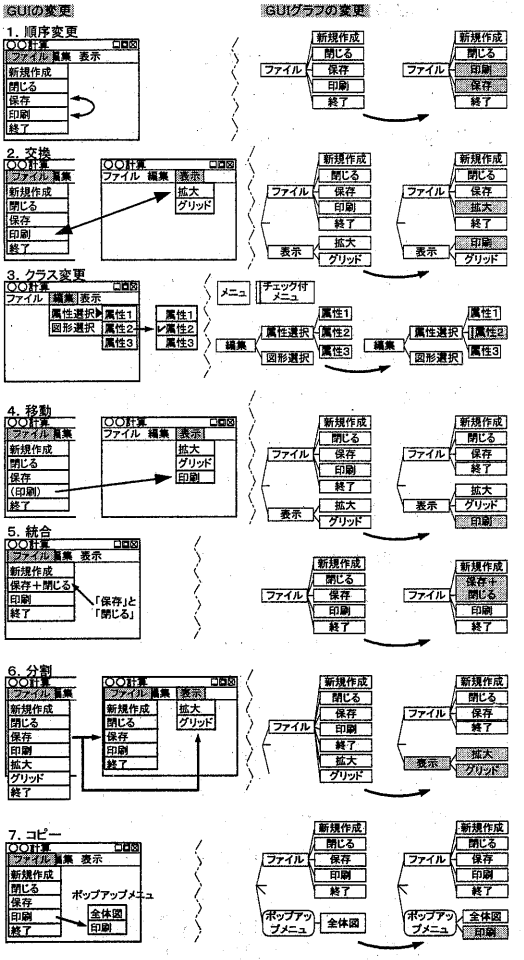

図 3: GUI 変更オペレーション (メニュを例として)

2. 移動不可： 初期状態から移動、変更しない

3. サイズ/容量制限: メニュやダイアログ内の GUI 要素数の上下限。

他アプリケーションを参照し、その GUI をテンプレートとして、初期構造や制約条件とすることも考えられる。ユーザが複数のアプリケーションを利用している場合、使い慣れた GUI 構成を、一部でも共通で利用できれば、操作性は向上すると考える。このようなテンプレートの適用は、変更オペレーションを実行する上で、GUI 要素の初期配置やグループ化などの制約条件として反映すればよい。

4.3 評価指標

構造の評価指標は、ユーザビリティ評価 (例えば [10]) の視点から、下記のようなものが挙げられる。

1. 総操作回数： 使用頻度の多いものを優先的に前/上に、連続して使うものは近くにすることで総操作回数を少なくできる。
2. デバイス切替： マウス入力とキーボード入力の切替えを少なくすることで使いやすくする。
3. マウス操作効率： 目標 GUI までの画面上での移動距離や GUI のサイズを調整することで操作を効率化できる (ex. フィッツの法則)

本研究では、最初の試みとして、GUI 要素として「メニュ」を対象にした。以下、「メニュ」を対象にした場合の、具体例を示す。

4.4 メニュ構造の再構成

メニュ構造の評価指標として下記を用いた。

> 「ログとして保存した複数の操作系列に対して、操作回数の総和を小さく」+「メニュアイテム数のばらつきを小さく (構造的に適度な分布)」

ここで第 1 項に対する操作回数のカウント法の例を、図 4 に示す。操作系列に含まれる各イベントメニュ間の移動距離を、連続したものとしてカウントする。また第 2 項の 'ばらつき' は、メニュアイテム数の分散を指標とした。これらを数式で表現すると、式 (1,2) となる。この値 E を "小さく" するような配置を求めることがメニュ再構成の目的である。

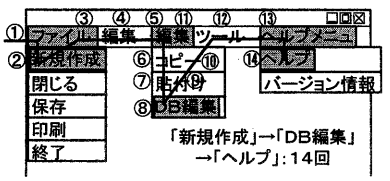

図 4: 操作回数のカウント法の例

$$E = \alpha E_1 + (1-\alpha) E_2 = \alpha \frac{1}{K} \sum_{i=1}^{K} \frac{c_i}{MaxC_i} \quad (1)$$

$$+ \quad (1-\alpha) \left\{ \frac{1}{N} \sum_{j=1}^{N} \left(m_j - \frac{M}{N} \right)^2 \right\} \Big/ \left\{ \frac{M^2}{N^2} (N-1) \right\}$$

$$MaxC_i = \sum_{i=1}^{e_i} (N + M - 1) \quad (2)$$

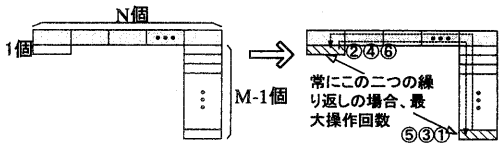

図 5: 操作回数が最大となる配置

ここで、K はイベント系列の総数、c_i はイベント系列 i に対する総操作回数、$MaxC_i$ はイベント系列 i に対する最大操作回数、N はタイトルメニュ数、M はアイテム総数、m_j はメニュ j のアイテム数、α は任意の定数。左辺 E_1 の $MaxC_i$ は、e_i をイベント系列 i のイベント数として、図 5 のような場合に最大となるのでそのときの回数である。また、タイトルメニュをまたがる操作のときは、タイトルメニュのカウント数に重みをつける (ex. タイトルメニュの一回の通過を数回分としてカウントする) ことも考えられる。これは、連続して利用されるアイテムを、同一ないし近傍のメニュ内に配置するように作用する。さらに、アイテム数が上下限値を超えた場合はその数に応じて評価指標が "大きく" なるようなペナルティー項を E に加えることも考えられる。

4.5 GUI 変更オペレーションの反復

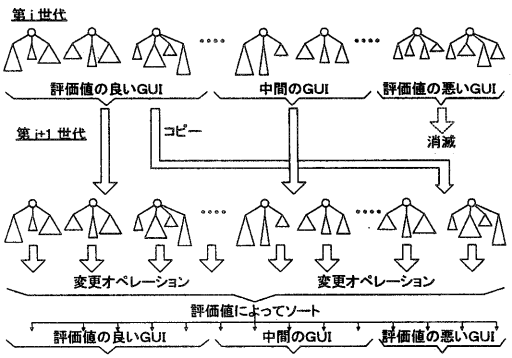

図 6: 反復計算の回数更新 (世代交代) の処理の例

GUI 変更オペレーションの反復は、基本的に通常の GP の計算に準じ、例えば、下記の通りとする。

- 反復計算の回数更新 (世代交代) のときは、エリート戦略に準ずる方法 (図 6) をとる。つまり、優秀な個体 (GUI) はそのまま残し、悪い個体は消滅させ、代わりに優秀な個体を "コピー" する。残りはそのまま残す (比率は任意に設定)。

4.6 シミュレーション

簡単なミュレーションの結果を示す。

4.6.1 シミュレーション時の制約条件

- 制約なし
- グループ制約
- グループ制約＋各グループをメニュに設定 (1 グループ= 1 メニュ)
- グループ制約＋テンプレート適用 (テンプレートのメニュ分割に準じて、グループをメニュとして初期条件に設定)

ここで、「制約なし」は全く制約なし、「グループ制約」は人間がアイテムのグループ化を行ったもの、「グループ制約＋ 1 グループ 1 メニュ」は、全アイテムのグループ化を行い、各グループをメニュ 1 つに対応付けたもの、「テンプレート制約＋グループ制約」は、他ソフトの GUI をテンプレートとして制約とし、さらに人間がグループ化の制約を与えたものである。

4.6.2 シミュレーション結果

同一の GUI 構造 (一つのメニュにアイテムを全部まとめたもの) から、同一の操作履歴を用いて、個体数を 100、計算の最大反復数を 500、としてシミュレーションを行った。結果を表 1 に示す。ここで総メニュアイテム数は 49 であった。

表 1: シミュレーション結果 (平均値)

条件	総操作回数
制約なし	782.5
グループ制約	783.5
グループ制約＋ 1 グループ 1 メニュ	**598**
テンプレート制約＋グループ制約	703
人の設計	812

シミュレーションから下記の結果を得た。

- 今回定めた「操作回数」という指標について、人の設計よりも良い結果が得られた。
- 移動頻度の (比較的) 多いメニュは隣接して配置された。

4.7 今後の展開

下記の観点から追加検討を行う予定である。

- メニュ以外への GUI 要素への適用
- 他の定量化可能な評価指標/制約条件の検討
- 操作履歴の解析による制約条件の自動設定
- GUI 設計への適用 (例 : ある作業フローが多い顧客に対しては、どのような GUI 構成がよいかを検討する)

5. オンラインでの予測 GUI の作成

操作履歴に基づいたオンライン適応機能として、「ツールバーボタンのオンライン変更」による操作支援と、「入力および選択値のデフォルトの変更」による操作支援、および「短縮操作ダイアログの自動生成」による操作支援について検討した。これらの機能についても、GUI グラフの更新処理で実現している。以下、それぞれについて説明する。

5.1 ツールバーボタンのオンライン変更

これは、直前の操作コマンドから、次のコマンドを予測し、複数の候補をツールバー上のボタンに動的に割り当てるものである。

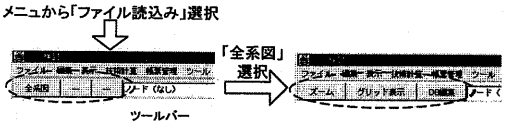

図 7: ツールバーボタンによる操作予測例

図 7 に例を示す。この例では、メニュから「ファイル読込み」を選択すると、過去の操作履歴から、次に操作されるであろうコマンドを予測し、その候補として、「全系図」機能がツールバーのボタンに割り当てられた。さらに続いて「全系図」のボタンを選択すると、次の操作の候補である「ズーム」、「グリッド表示」、「DB 編集」機能がツールバーボタンに割当てられている。これは、図 3 に示した“コピー”機能により、予測された次の操作に対する GUI の情報 (属性や機能) を、ツールバーの部分木のボタンそれぞれに、上書きコピーして実現する。

5.1.1 コマンド予測手法

コマンドの予測には様々な手法が考えられるが、現状は以下のような単純な手法で計算している。

1. 頻度 : あるコマンドが実行された後に、過去に多く実行された回数順に候補とする

2. 時間重み付の頻度：$F_t(c,a)$ の降順に候補とする (式 (3))。

$$F_t(c_j,a) = \frac{\sum_{k=1}^{T} \rho(T-t_k) g(c_j,a,t_k)}{\sum_{i=1}^{M}\sum_{k=1}^{T} \rho(T-t_i) g(c_i,a,t_k)} \quad (3)$$

ここで T はタイムスタンプ (ex. 時刻、コマンドステップ数) の現在値、M はコマンド総数、t_k はタイムスタンプ、$g(c_i,a,t_k)$ はコマンド a が実行されたあとに、タイムスタンプ t_k でコマンド c_i が実行された回数 (実質は 1 か 0)、$\rho()$ は任意の単調減少関数である。同じ頻度だけ実行されたコマンドであっても、過去に実行されたものほど優先度が減り、最近に実行されたものほど優先度が上がる。

5.2 選択値、入力値などの自動/デフォルト設定

リスト、チェックボックス、テキストフィールド、コンボボックスなどについては、次回のデフォルトに自動で変更することが有効である。またテキスト入力に関しては、過去に入力した複数の文字列から選択リスト (コンボボックス) を自動生成することも有効である。これらはXML記述では、部分ツリーの変更や属性の変更 (GUI変更オペレーションの「クラス・属性変更」) などによって実現する。これらの変更によって、特に繰返し作業の効率が向上すると考える。

例を図8に示す。上から順に、下記の通りである。

- リスト：前回の選択値が自動選択
- チェックボックス：前回の選択値が自動選択
- テキスト入力：前回の入力値が自動入力
- テキスト入力：これまでに入力した文字列からコンボボックス (選択リスト) を自動作成。
- コンボボックス：前回の選択値が自動選択

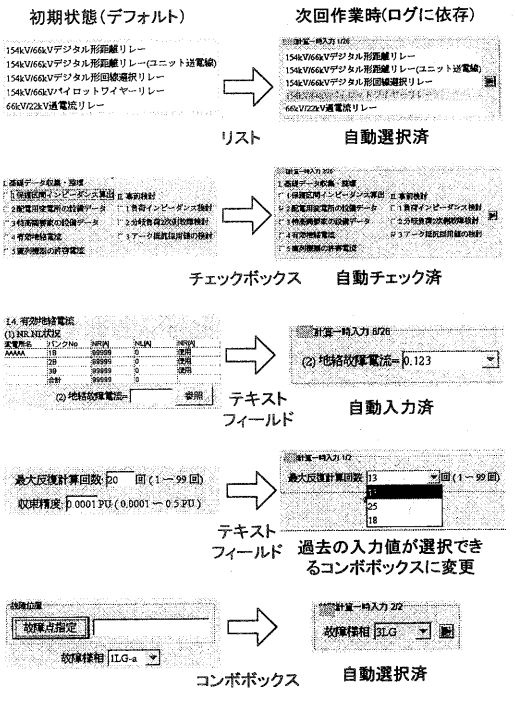

図 8: 選択値、入力値などの自動/デフォルト設定例

5.3 短縮操作用ダイアログの自動作成

連続して開かれる複数のダイアログを利用した入力作業について、過去に利用されたGUI要素からなるダイアログを、自動作成することを検討した。ダイアログを順次開き、入力を繰返し利用する作業について、効率化が図れると考える。

作成方法の概要を図9に示す。まず、操作履歴から利用されたGUI要素を認識し、XMLツリーから

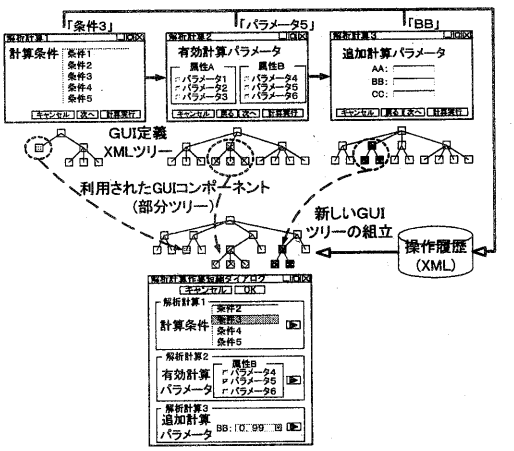

図 9: 短縮操作用ダイアログ作成の概要図

該当する部分ツリーを検索する。そして、それらを統合して新たなダイアログのXMLツリーを作成する。次回、作業を行うときは、複数のダイアログを開いて順次入力を行う必要がなく、一つのダイアログ内で作業ができる。また、前節で示したデフォル

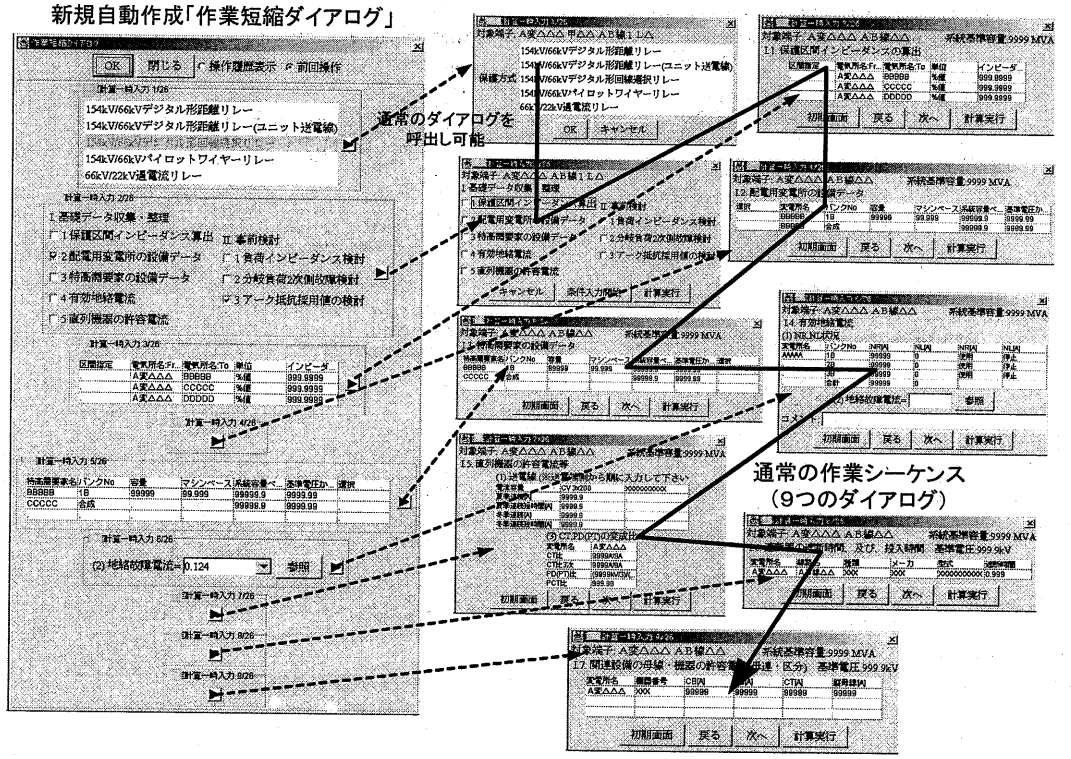

図 10: 短縮操作用ダイアログの生成例

ト設定機能により、同じ作業を行う場合は、再入力の必要はない。

作成例を、図 10 に示す。この例は、9 つのダイアログからなる作業シーケンスを、一画面で操作する「作業短縮ダイアログ」を示している。また、ダイアログ内の右矢印ボタンを押せば、その作業入力に対応したオリジナルのダイアログを開くことができ、過去、入力や選択しなかった GUI 要素については、ここで操作が可能となる。

連続して開かれるダイアログとは別に、同一のダイアログを利用して、データの編集や確認を行う作業は様々な場面で登場する。上記と同様の技術によって、データの編集に利用した GUI 要素からなるダイアログを、自動生成することも可能である。一度編集したデータについて、繰返し確認や編集をする作業に対して、効果を発揮すると考える。

作成方法は、図 9 に示した手法と基本的に同様である。まず、操作履歴から利用された GUI 要素とそのときに編集されたデータ ID を認識し、XML ツリーから該当する部分ツリーを検索する。そして、それらを統合して新しいダイアログの XML ツリーを作成する。次回、作業を行うときは、例えば画面上で各データを逐一指定して編集画面を開く必要がなく、一つのダイアログ内で編集、確認作業が行える。

作成例を図 11 に示す。この例では、3 つのデータの編集画面の合成を示している。「複数のデータの、あるパラメータを微調整しながら計算を繰り返す」などの作業に対して、左上の新ダイアログが効果を発揮する。

5.4 今後の展開

下記の観点から追加検討を行う予定である。

- 予測アルゴリズムの高度化
- 新しいツリーを合成する際の、
  - GUI の粒度の自動設定
  - GUI 要素のレイアウト手法

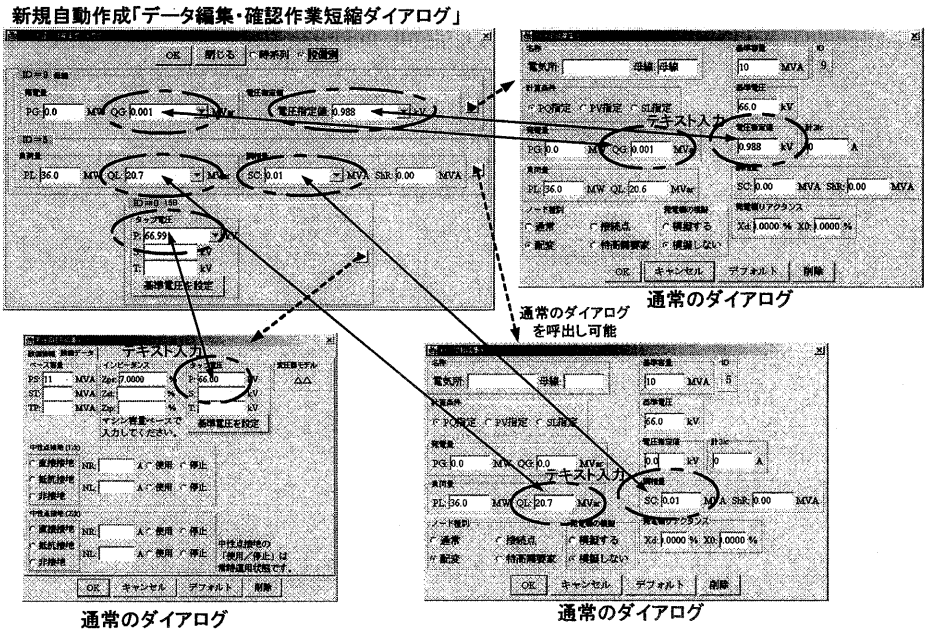

図 11: データ編集・確認作業短縮ダイアログの生成例

6. その他

下記の変更についても、XML で統一的に実現可能と考える。これらは主に、XML ツリーにおける、各ノードの属性変更で実現すればよい。

1. GUI 要素の前景・背景色
2. GUI 要素のサイズ
3. GUI 要素内の文字フォント
4. GUI 要素の表示位置

現実の利用場面では、例えばこれらの属性変更を XSL (Extensible Stylesheet Language) によるスタイルとして記述しておき、アプリケーションを実行する際に、GUI を更新するように実現してもよい。これらは適応とは異なるが、Look & Feel やスキンの選択と同様に、ユーザの好みに応じるものである。

7. おわりに

本稿では、XML 記述した GUI グラフを、ユーザの操作履歴に基づいて、オフラインおよびオンライン適応させる試みについて説明した。現在、オフライン適応については特に効果検証を、オンライン適応については、特に予測アルゴリズムの高度化を検討中である。

XML はコンテンツ記述の標準仕様として、急速に広まりつつあるが、今回示したように、コンテンツ以外の記述にも強力なフォーマットである。本稿で示した技術を、ネットサービスの画面や携帯電話/端末の画面などへの適用も試みたいと考えている。

参考文献

[1] 増井、"適応型インタフェース"、UNIX MAGAZINE, 1998.11, pp.181 (1998)

[2] 増井、"予測/例示インタフェースの研究動向", コンピュータソフトウェア, Vol.14, No. 3, pp.220 (1997)

[3] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, K. Rommelse, "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users", Proc. of UAI-98 (1998)

[4] 増井、"ペンを用いた高速文章入力手法", 日本ソフトウェア科学会 WISS'97, pp.51 (1997)

[5] 寺岡、京本、押田、秋吉、"ユーザの操作履歴に基づく GUI 適応化の試み"、第 61 回 情処全大、3L-3 (2000).

[6] http://www.alphaworks.ibm.com/tech/bml/

[7] http://uiml.org/

[8] http://java.sun.com/products/jfc/

[9] 伊庭、"遺伝的プログラミング", 東京電機大学出版局 (1996)

[10] 黒須、伊藤、時津、"ユーザ工学入門"、共立出版 (1999)