インタラクティブプログラムのための操作履歴視覚化手法

Visualizing User Operation History in an Interactive Program

中村 利雄　　五十嵐 健夫 *

Summary.　本研究では，ユーザのインタラクションに基づいてプログラム操作履歴を視覚化する手法を提案する．本システムは画面のスナップショットから構築される静止画の上に一連の連続するユーザ操作を注釈付けていくことで概略ストーリーボードを自動的に生成する．この概略ストーリーボードの利点は動画のように再生作業を伴うことなく操作の概要と捉えることができることである．全てのアプリケーションで最も効率良く操作履歴を視覚化する仕組みを定義することは不可能であるが，注釈内容や視覚化の粒度を動的に切り替え可能としていくことで操作履歴をより効果的に視覚化することができると考えられる．

1　はじめに

過去にユーザが行った操作を再利用することで利便性を高めようとする試みが数多く行われている．例えば，ユーザが直前に行った操作を取消/再実行する Undo/Redo 機能，過去に実行した操作を繰り返すヒストリ機能，操作内容を汎化してプログラムを生成する例示プログラミング，また使用頻度や行動パターンを解析して特徴を抽出するといったものがある．これらの機能を有効に利用するには，膨大な量の操作履歴を効率良く理解して目的とする状態を素早く認識できることが重要である．

一方で，操作履歴の提示方法については依然としてテキスト形式が主流である．テキスト形式は扱いやすい反面，図形や動きなどの視覚的情報を伝えづらいという問題がある．スナップショットを時系列に並べて提示する方法 [4, 5] もあるが，スナップショット間の操作内容については画像とテキストからユーザ自身が頭の中で連想しなければならない．さらに GUI 操作を伴うアプリケーションではユーザのインタラクションに強く依存して動作するため，「操作の過程」も重要な意味を持つ．そのため，プログラムの操作履歴とその時の実行状態を直感的イメージとして認識するのは困難である．

本研究では，GUI アプリケーションに特有である「ユーザとのインタラクション」を概略ストーリーボードを用いて視覚化する手法を提案する．システムは記録された操作イベント列を解析して意味操作を抽出する．そして，これらの意味操作をスナップショットから構築した背景画像上に注釈付けていくことで概略ストーリーボードを生成する (図 1)．この概略ストーリーボード表現はスケッチやジェスチャのような動きのある一連の操作を 1 枚の静止画として表現することが可能であり，ユーザが操作履歴を理解するのを視覚的に支援する．また，概略ストーリーボードで表現された各シーンの詳細を理解するため，必要に応じてシーンの実行状態を実際に再現させる機能も提供している．

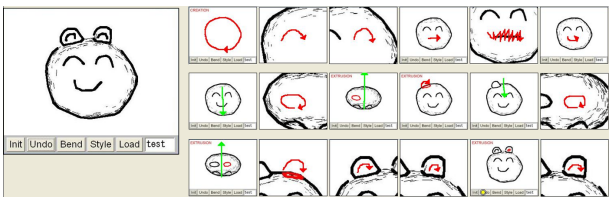

図 1. 操作履歴から生成される概略ストーリーボード

2　関連研究

ユーザが過去に行った GUI 操作をグラフィカルな形で提示するシステムとして Chimera[4] や Mondrian[5] などがある．これらはグラフィックス編集を対象としたシステムであり，Chimera ではオブジェクトのスナップショットを紙芝居風に並べることで，さらに Mondrian では図形の操作前と操作後のスナップショットを組にして表現することで図形の編集操作履歴を提示している．どちらのシステムも図形が編集される過程は把握できるが，提示される画像間の動きや遷移に関しては画像と操作を表すテキストから連想しなければならない．また，表現されるのはあくまで画面のスナップショットであり実際のユーザが行う操作を直接は表現していない．

画面上の視覚変化を残像やフラッシュバックなどの動画を用いて効果的に理解支援をする手法 [2] が提案されている．この手法は視覚変化の概略を短時間で捕捉するのに適しているが，一度に複数の箇所で視覚変化が起こるとユーザの認識力が低下してしまう．また，Phosphor[1] では GUI 操作の残像を画面に一定時間残すことでユーザの操作支援を行おう

* Toshio Nakamura, 東京大学大学院 情報理工学系研究科 コンピュータ科学専攻, Takeo Igarashi, 東京大学大学院 情報理工学系研究科 コンピュータ科学専攻 / JST SORST

と試みている．これはユーザが過去にどのような操作をしてきたか理解する大きな助けとなるが，実際に残像として残るのはGUIの画面変化でありユーザの操作を直接は表現していない．大量の操作を全て残像に残した場合，操作の過程を把握するのが困難となる問題もある．本システムでは，ユーザの操作を注釈付ける形で静止画として概略画像を提供する．

一方で，概略を生成するという試みは映像の分野で多く研究されている．特に関連が強いものとしては [3] が挙げられる．この論文では物体の動きやカメラワークを3Dの矢印を用いて効果的に表現しようとしている．本研究はユーザのインタラクションに注目し，この考えを応用することでプログラム操作履歴をより効果的に視覚化しようと試みている．

3 操作履歴の視覚化

ユーザのインタラクション操作を視覚化する方法として，操作中のスクリーン画面を動画として記録して観察する方法が挙げられる．これはプログラム操作中で実際に目にする光景のため正確に状況を伝達可能でかつ理解もし易いが，動画を見る行為は時間のかかる作業である．たとえ早送り等で再生して時間を短縮しようとしても，逆に詳細な変化を認識するのが困難となる．さらには，再生させなければ代表画像以外の情報は得られないという欠点もある．

これらの問題も考慮して，本研究では映写スライド形式を用いて一連の連続したユーザ操作を説明する概略ストーリーボードを提示する．概略ストーリーボードとは映画製作やコミックなどでよく使われる表現であり，吹き出しや矢印などを用いることで動きや状況変化といった時間の継起性を静止画として一度に表現することができる．

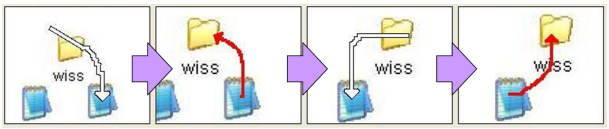

図 2. 「2つのファイルを右，左の順でフォルダにドラッグ&ドロップする」という操作の視覚化

図2は概略ストーリーボードによる視覚化の例である．この例では，矢印によりマウスポインタの軌跡 (白:マウス移動，赤:ドラッグ) を表している．このように1枚の画像上に意味のある一連の連続するユーザ操作を図やラベル等で注釈付けていくことで操作履歴を視覚化する．マウス移動などある種の操作が意味を持たず不要であったり1枚の静止画に注釈付ける適切な数は視覚化するアプリケーションやその時の目的により異なるが，必要に応じてフィルタリングやクラスタリングなどの処理をカスタマイズしていくことでより効果的かつ汎用的な概略ストーリーボードを生成することができる．

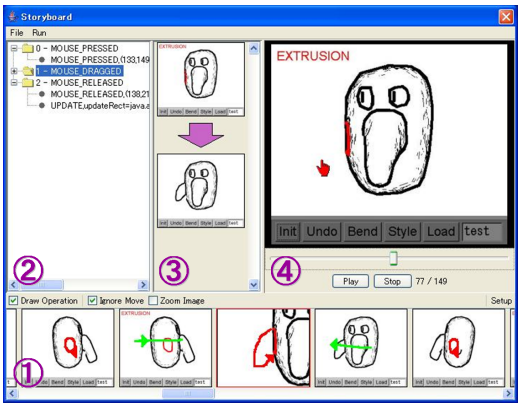

図 3. システム概観：1. 概略ストーリーボード，2. イベント一覧，3. サムネイル一覧，4. プレビュー画面

4 ユーザインタフェース

本システムの概観を図3に示す．画面は2つの領域に分けられており，下部は操作履歴の概略を表示する概略ストーリーボード領域，上部は注目する操作履歴の詳細を確認するための領域であり，左からイベント一覧領域，サムネイル一覧領域，プレビュー画面領域である．

ユーザはまず概略ストーリーボードを閲覧して操作履歴の概略を捉える．各シーンは時間軸に沿って紙芝居風に並べられているので，注目したシーンが履歴中のどの場所に位置した操作なのかを容易に把握することができる．また，通常では各シーンは操作対象となるウィンドウ全体を表示しているが，ユーザが操作している部分領域のみを拡大表示するよう切り替えることも個別に設定可能である．

概略ストーリーボードからシーンを選択すると，そのシーン内で記録されたイベント系列と画面のスナップショットがそれぞれイベント一覧とサムネイル一覧の領域に表示される．イベント系列は連続する同一種類のイベントを木構造でまとめられており，葉が実際に記録されたイベントとなる．サムネイル一覧では操作中での画面変化を確認することができ，特に最初と最後のサムネイルはシーン内での操作の直前と直後のスナップショットに対応する．また，プレビュー画面ではそのシーンの具体的な様子をアニメーションで再生することが可能である．この際，操作対象となっているGUIコンポーネントはハイライトされる．さらに，マウス操作に関しては実際のマウスではなく，色付けされた擬似マウスによりドラッグやクリック等の状態が区別できるようにようになっている．

最後に，本システムは概略ストーリーボードにある各シーンのプログラム実行状態を実際に再現する機能を持つ．対象シーンを選択した状態でメニューから再現命令を実行すると，システムは自動的にそのシーンの実行状態を再現する．実行状態を再現し

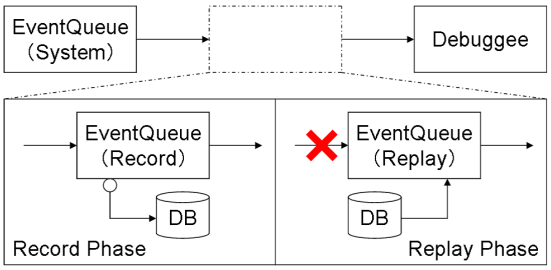

図 4. システムの記録/再生処理モデル

た後，ユーザは引き続きアプリケーションを操作したり再度別の実行状態を再現させることも可能である．

5 実装

本システムは Java を用いて実装されている．対象は AWT/Swing を使用した Java アプリケーションであり，これらのアプリケーション上でのユーザ操作履歴を視覚化する．本システムの構成は主にユーザの操作データを記録する記録処理部，記録した操作データに基づいて視覚化処理を行う概略ストーリーボード生成部，そして過去の実行状態を再現する再現処理部から成る．

5.1 記録処理部/再現処理部

Java における GUI へのインタラクション処理は EventQueue(SystemQueue) で管理され，各操作に対応するイベントを対象アプリケーションへ送ることで処理が行われている．そこで本システムでは専用の EventQueue(RecordQueue，ReplayQueue) を導入し，これらを適宜 SystemQueue に取り付けるで操作イベントの記録処理と実行状態の再現処理を実現している (図 4)．そのため，ユーザはアプリケーションに一切変更を加えることなく本システムを利用することができる．

記録処理では RecordQueue を取り付け，通常の処理と並行してイベントの記録を行う．だたし，状態を再現させる時の必要性からマウス操作やキー入力などの操作イベント以外にも画面描画やフォーカス変更，ウィンドウやコンポーネントに関する各種通知イベントといった EventQueue からアプリケーションに送られるイベント全てを記録している．記録が必要ない場合には RecordQueue を取り外すだけでよい．また，必要な時 (例えば，Mouse Press や Mouse Release の前後) にはイベントの記録と並行して操作対象であるウィンドウのスナップショットも取得する．一方，再現処理では ReplayQueue に切り替え，アプリケーションを初期化した後，SystemQueue から送られてくるイベントをブロックしながら記録したイベントを順次アプリケーションへ送ることで任意地点での実行状態を再現する．

図 5. 各変換処理にて抽出される操作例

なお，現在の実装では GUI に関するイベントしか記録していないため，それ以外の要素 (ネットワークやタイマーなど) によりシステムの内部状態が変化するアプリケーションには対応できない．

5.2 概略ストーリーボード生成部

このフェーズでは記録処理部で取得した生のイベント系列と画面のスナップショットから概略ストーリーボードを生成する．

5.2.1 操作解析処理

操作解析処理ではイベント系列から意味的にまとまりのある一連の操作を抽出する．記録処理部で取得された生のイベント系列は操作イベント (マウス操作やキー入力など) 以外にも様々なイベントが混在している．さらに，操作イベントのみに注目してもそれらは一連の操作を表すイベントではなく各々が事象を説明するための独立したイベントとして構成されている．そこで図 5 に示す 3 つの変換処理を適用することで，低レベルのイベント列からより高レベルである意味イベントを抽出していく．

最初の変換処理は操作イベント以外のイベントの畳み込み処理である (図 5(a))．描画イベントやウィンドウイベントなどはユーザの操作を表したイベントではなく，アプリケーションが内部での処理用に生成するイベントである．従って，これらのイベントは直前の操作イベントを処理した結果発生したものであると考え，直前の操作イベントに基づいて新たなイベントへと変換する．本システムでは，このイベントを基本操作イベントと定義する．

変換された基本操作イベントには同じ事象を表すイベントが連続する箇所が多数存在する．そこで次の変換処理では連続して出現する同タイプのイベント系列を抽出し，これを新たに繰り返しイベントとして変換する (図 5(b))．ただし，マウスが GUI コンポーネントの境界に出入りしたことを表すイベント (Enter と Exit) に関しては，マウスが移動した

結果起こる事象なのでマウス移動と同タイプのイベントとして扱うことにする．

最後の変換処理ではイベント系列から意味を持つ操作の抽出を行う (図 5(c))．例えば，マウスのクリック操作はマウスを押して放すという一連の動作の結果である．このように一連の意味を持った操作イベント列を意味イベントとして変換していく．この意味イベントはアプリケーションとしては低レベルであるが，非常に汎用的であり後続する処理においてベースとなるものである．

5.2.2 フィルタ/クラスタ処理

ここでは操作解析処理により抽出された意味イベント列に基づいて，より高レベルで意味のある操作へと変換していく．具体的には，視覚化する必要のない操作をフィルタリングしたり，複数の操作を 1 つの静止画に描画するためにクラスタリングを行う．

一般的に，これらはアプリケーションやユーザの目的によって多種多様となるので一意に決定することは不可能である．例えば,「ほとんどのアプリケーションではマウスの移動操作は全く処理されないので視覚化する必要がないが，あるアプリケーションではマウスが GUI コンポーネントの境界をまたぐ際に何らかの処理がされるので視覚化したい」，また別の例としては「パネル上をダブルクリックするとプロパティが表示されるので，ダブルクリックを一つの操作としてクラスタリングしたい」などである．

これらの要求に対応するため，アプリケーション毎に処理モジュールを付加的に組み込むことでフィルタ/クラスタ処理を行う．現在のプロトタイプでは，この処理モジュールを事前にプログラムとして作成して手動で組み込む設定となっているが，今後の課題としてファイルや Programming by Demonstration などを用いて動的に指定・変更できるようにしたい．

5.2.3 シーン生成処理

シーン生成処理ではフィルタ/クラスタ処理された操作履歴から概略ストーリーボードの要素となるシーンを生成する．まず，シーン中に取得された画面のスナップショットからシーンを表す背景画像を構築していく．一般に何らかの操作を実行するとアプリケーションは結果をグラフィカルな形で表示するため，背景画像はこれらを考慮して構築されるのが理想的である．しかし，現状は単にそのシーンの先頭画像を選択する実装になっている．その後，各操作に対応した注釈を構築された背景画像上に描画していくことで概略ストーリーボードを生成する．

6 まとめと今後の課題

本研究では，ユーザのインタラクションに基づいてプログラム操作履歴を概略ストーリーボードの形で視覚化する手法を提案した．本システムは動画ではなく概略を説明した静止画として操作の過程を視覚化する．これにより，ドラッグ&ドロップなど時間を伴う操作も一度に表現することが可能となり，直感的イメージとして実行状態を捉えることができる．

今後の課題としては主に以下の 3 つが挙げられる．

第 1 はユーザ操作の抽出処理やフィルタリング/クラスタリングの手法である．特殊な場合を除き，GUI は各コンポーネント毎に対象とする操作が決まっているものが多い．例えば，ボタンではクリック操作に意味を持つが，スクロールバーではドラッグ操作に意味がある．しかし，現状では操作の対象となる GUI については考慮せずクリックやドラッグといった操作の種類のみで処理している．これらを考慮していくことでより意味のある操作が抽出可能になり，かつフィルタ/クラスタ処理を処理モジュールが無くてもある程度行うことができる．

第 2 は概略ストーリーボードの表現力向上である．現在の実装では各シーンの背景となる静止画は単にシーンの先頭画像を用いただけである．しかし，ファイルのドラッグ&ドロップ操作などは一連の操作中も動的に状態を変化させ結果を更新するため，操作中の変化をユーザが連想するか詳細画面で確認する必要がある．従って，操作中におけるオブジェクトの動きといったアプリケーションの外観変化も静止画で効果的に表現する手法を色々と検討していく．

第 3 はユーザスタディによる評価である．何人かのユーザに使ってもらい意見を貰ったが，アプリケーションの種類により有用性は異なるはずである．今後は図形エディタや表計算ソフト，ゲームなど様々なアプリケーションで詳細な評価実験を行い，既存の視覚化手法と比較して本手法がどの程度有効であるかを調査していく予定である．

参考文献

[1] P. Baudisch, D. Tan, M. Collomb, D. Robbins, K. Hinckley, M. Agrawala, S. Zhao, and G. Ramos. Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects. In *Proceedings of UIST*, pp. 169–178, 2006.

[2] A. Bezerianos, P. Dragicevic, and R. Balakrishnan. Mnemonic rendering: An Image-Based Approach for Exposing Hidden Changes in Dynamic Displays. In *Proceedings of UIST*, pp. 159–168, 2006.

[3] D. B. Goldman, B. Curless, D. Salesin, and S. M. Seitz. Schematic Storyboarding for Video Visualization and Editing. In *Proceedings of SIGGRAPH*, pp. 862–871, 2006.

[4] D. Kurlander. Chimera: Example-Based Graphical Editing. In *Watch What I Do: Programming by Demonstration*, pp. 271–290. MIT Press, 1993.

[5] H. Lieberman. Mondrian: A Teachable Graphical Editor. In *Watch What I Do: Programming by Demonstration*, pp. 341–358. MIT Press, 1993.