ソフトウェアテストシンポジウム 2003
JaSST'03: Japan Symposium on Software Testing 2003

ダイナミックに操作を補完するテスト自動化について

小坂 史† 山口 聡之†

†富士ゼロックス株式会社 商品評価部 システム検証センター

〒213-8508 神奈川県川崎市高津区坂戸 3-2-1 KSP R&D
E-mail: †{Fumi.Kosaka,Akinobu.Yamaguchi}@fujixerox.co.jp

あらまし GUI をともなうソフトウェアのテスト自動化において，これまで記録再生方式によりテストケースを作成し実行する手法が用いられてきた．しかし，GUI の複雑化により，状態遷移情報が膨大になり，すべての状態遷移の把握が困難になってきている．また状態遷移情報は仕様として記述されていないケースがある．本研究では GUI の構造に着目し，最終的に設定したい操作とその設定に至るまでの操作を分離する．最終的な設定に至るまでの操作として補完操作をあらたに定義してダイナミックに操作を補完しながらテストの自動実行をおこなう手法について述べる．

キーワード ソフトウェアテスト，テスト自動実行，補完操作, Windows, GUI, 状態遷移, pre- and postconditions

On the test automation which complements operation dynamically

Fumi KOSAKA† and Akinobu YAMAGUCHI†

† System Verification Center , Customer Focus and Quality Unit, Fuji Xerox Co., Ltd.

3-2-1,Sakado, Takatsu-ku Kawasaki-shi, Kanagawa, 213-8508, Japan
E-mail: †{ Fumi.Kosaka,Akinobu.Yamaguchi}@fujixerox.co.jp

Abstract In test automation of the software with GUI, the technique of creating and performing a test case by capture and replay has so far been used. However, due to complication of GUI, state conditions increases and to grasp all state conditions is becoming more and more difficult. Moreover, in some cases, state conditions information is insufficient as a specification. In this paper, we divide operations into a operation to set up finally and operations which are necessary to result in the final operation by paying attention to the structure of GUI. We define a supplement operation as the operation to cause a final setup and describe the technique of performing automatic execution while assembling a sequence of complement operations dynamically.

Keywords software testing, automated test case execution, supplement operation, Windows, GUI, state transition, pre- and postconditions

1. はじめに

ソフトウェアの開発期間が短縮される中，テスト自動化への要求が高まってきている．記録再生方式を利用した GUI をともなうソフトウェアのテスト自動化は記録した操作手順でしか再生できないため，すべてのテストケースについて予め膨大な手順を入力するか，もしくはプログラミングする必要がある．ソフトウェアの GUI 部の構造情報および状態遷移情報をあらかじめ作成しておき，その情報から網羅的にテストケースを作成して実行するソフトウェアのシステムテスト装置の提案がこれまでなされてきた [1]．しかし，ウィンドウの配置や種類との組合せ，動的に変化するコントロールなど GUI が複雑になるにつれて，状態遷移情報が膨大になり，すべての状態遷移の把握が困難になってきていること．また，状態遷移情報のすべてが仕様書に記述されていることは稀であり，状態遷移情報を網羅的に記述することは非常に困難である．さらに，それらの情報はテスト途中での仕様変更により自動化のためのデータの作り直しが発生する．テスト設計者がテストオペレータへ指示するテスト作業指示書には，通常最終的に設定すべきテスト手順が記述されている．しかし，その設定するまでの具体的な操作手順については記述されていない．そこで本研究では，ソフトウェアの GUI 構造に着目し，(1)必要な状態遷移の情報を少なく，(2)最終的に設定すべき操作とその操作に至るまでの操作を分離，(3)最終的に設定すべき操作を選択して実行指示するだけで，自動的に必要な操作を検索してダイナミックに補完しながら実行するソフトウェアを実装したので紹介する．これにより，「最終的に設定すべき操作」と「補完操作」を分離できるので，仕

様変更のデータの作り直し作業を減らし，何回も繰り返しおこなわれるテスト[2]を効率よくおこなうことが可能となる．また，GUI を含むソフトウェアの自動化を GUI の仕様変更に対して影響を受けにくいデータ構造で容易におこなうことを可能にする．

本論文では，2.でソフトウェアのテストを自動化するうえで必要な情報について明確にする．3.で自動実行の処理について，いかにダイナミックに補完しながら実行していくかについて述べる．4.でそれを可能にするシステムについて説明する．5.で実際の製品評価における実行例を示す．6.で考察する．

2. テスト自動化において必要な情報

Windowsソフトウェアのテスト自動化において，状態を把握するために以下の情報が必要である．

- ウィンドウの状態
- ウィンドウへのコマンド

2.1. ウィンドウの状態

本研究では，Windows GUIのウィンドウの状態を4種類に分類した(図 1) .ここで述べるウィンドウとはボタンやメニュー等のGUIオブジェクトを指す．
ENABLEとはウィンドウが開いていて，可視であり，入力が有効な状態であることを示す.DISABLEとはウィンドウが開いていて，可視であり，入力無効な状態であることを示す．HIDDENとはウィンドウが開いていて，可視でない状態であることを示す． CLOSEとはウィンドウが閉じた状態であることを示す． また，Windowsの個々のウィンドウはコントロールとしての役割を持ったときの状態として次のようなものを持つ．

- コンボボックスのアイテムが選択されている
- リストボックスのアイテムが選択されている
- ツリーのアイテムが選択されている
- ラジオボタンが押されている
- チェックボタンがチェックON/OFFされている

2.2. ウィンドウへのコマンド

ウィンドウへの入力をコマンドと定義する．コマンドはWindows GUIに入力するオペレーション単位で，代表的なものを以下に示す．

- ボタンのクリック
- エディットボックスへの入力
- コンボボックスのアイテムの選択
- リストボックスのアイテムの選択
- ツリーアイテムの選択
- タブの選択
- メニューの選択

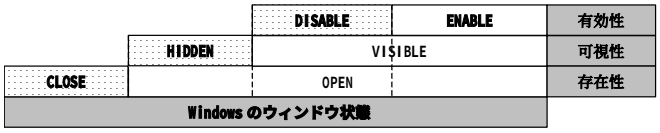

図 1 ウィンドウの状態

2.3. 操作

コマンドの実行によるGUIの状態の変化にはpre- and postconditionsを利用する．ひとつの操作は，コマンドを入力する前の状態であるpre-state,コマンドおよびコマンドを入力した後の状態であるpost-stateを持つ．操作には最終的に設定すべき操作であるユーザ操作とその設定に至るまでの操作である補完操作がある．最終的な設定に至るまでは複数の補完操作の流れがあり，それを補完シーケンスと呼ぶ．例えば，プリンタドライバのテスト指示書を考える．テスト設計者よりテストオペレータへ渡されるテスト指示書(表 1)には，最終的に設定すべきテスト手順は記述されているが，その設定をするまでの具体的な操作手順は記述されていない．しかし，この最終的な設定をおこなうに至るまでには,いくつかの補完操作(補完シーケンス)がある(表 2).

表 1 テスト指示書

設定	
アプリケーション	Word2002日本語版
テスト文書	testdocument.doc
部数	4
解像度(dpi)	600
出力用紙サイズ	B4
用紙の向き	よこ

表 2 補完操作

補完操作	
アプリケーションを起動する	Wordを実行
ファイル選択画面を開く	“ファイルを開く…”メニューを選択する
印刷画面を開く	“印刷…”メニューを選択する
プリンタドライバを開く	“プロパティ…”ボタンをクリックする
プリンタドライバを閉じる	“OK”ボタンをクリックする
タブのプロパティシートを切り替える	“タブ”をクリックする
アプリケーションを終了する	“終了”メニューを選択する

3. 自動実行の処理

自動実行の処理について以下にまとめる.

3.1. ダイナミックな操作補完

実行時には，指定されたユーザ操作を実行するために補完操作探索をおこない補完シーケンスを構築する.ユーザ操作が指定されたら,テスト対象のGUIの状態を調べる．あるユーザ操作(user seq)が指定されて，テスト対象のGUIの状態がaであったとする(図 2) .図 2において,指定されたユーザ操作はpre-stateとして状態cを要求している．このとき，状態aをpre-stateに持ち,状態cをpost-stateに持つ補完操作探索をおこない補完シーケンスを構築する(図 3)．現在は補完シーケンスは最短距離のものが選択され，補完シーケンスが同距離の場合は，先に見つかったものが選択されるように実装してある．また，同じpre-stateとpost-stateを持った補完操作はループしてしまうので存在してはいけないルールとしている．補完シーケンスが構築されたら，実行をおこなう．補完シーケンスは静的なものではなく，一つの操作が終了して状態が変化するたび補完操作探索がおこなわれダイナミックに再構成される．計画通りにテスト対象のGUIが変化していけば，図 4のように実行される.

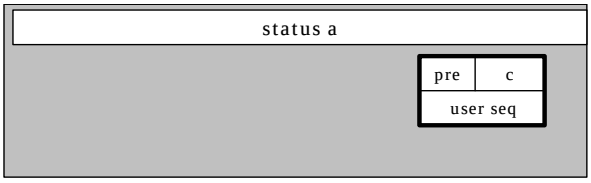

図 2 ユーザ操作

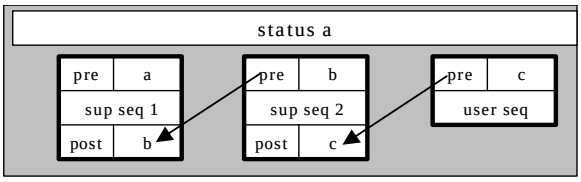

図 3 補完操作探索結果

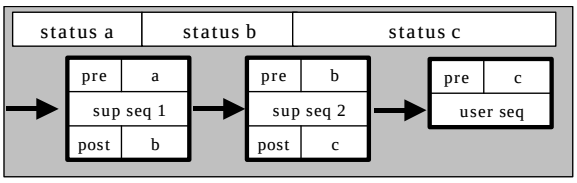

図 4 実行

実行の途中では，モーダルダイアログが上に表示されていたり，コンファーマが出現して補完操作探索のとおりには実行できないことが多い．モーダルダイアログが上に表示されていたり，コンファーマが出現して実行できない場合の対応については次節で述べる.

3.2. 暗黙の補完ルール

実行時に操作したいダイアログの上にモーダルダイアログが表示されている場合は，前節で説明したpre-statusとpost-statusがつながらず,対応できないケースとなる(図 5)．図 5のような場合は，操作したいコントロールウィンドウ(ボタン)の親ウィンドウでかつトップ階層であるウィンドウを探す．すると，ダイアログAであることがわかる．ダイアログAの状態がDISABLEであるためクリックできないので，3.1の補完ルールでは，post-stateでダイアログAがENABLEになり，pre-stateでダイアログAがDISABLEとなる補完操作探索をおこない補完シーケンスを構成しようとする．そのようなケースは非常に多くあり，すべてのケースの補完操作を記述することはできない.そこでこの様な場合には,トップ階層のウィンドウ(ダイアログA)がDISABLEの時は，上に表示されているウィンドウを検索し，それをCLOSEする補完シーケンスのパスを構築して実行するという暗黙の補完ルールを適用する(図 6).

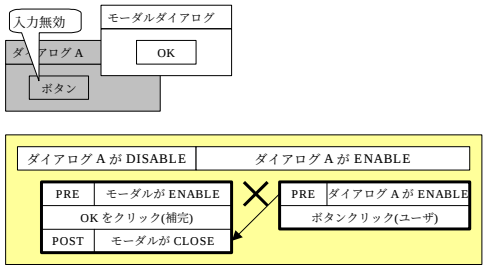

図 5 モーダルダイアログが被さっている場合

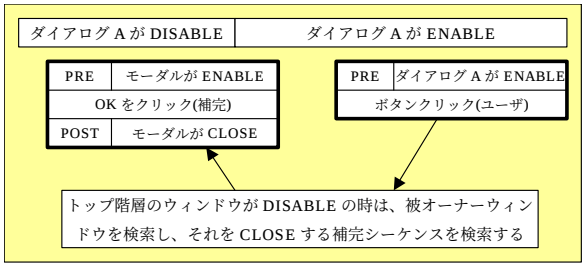

図 6 モーダルダイアログが上に表示されている場合の補完シーケンス

また,GUI情報としても取得していなく,補完操作として記述していない，エラーや確認のためのコンファーマについてもモーダルダイアログが上に表示されている場合と同様に考える．異なる点は，出現した時点でGUI情報を取得して，pre-stateとしてコンファーマがENABLE,post-stateとしてコンファーマがCLOSEとなる補完操作を動的に作成することである．作成した情報は，次回のテストのために記憶されて利用者に再利用される.補完操作の具体的なコマンド(コンファーマの対処方法)については利用者に問い合わせをおこなって決定する．

4. 自動実行システム

4.1. システムの概要

これまでのソフトウェアテスト自動化ツールは記録再生方式が中心であり，すべての状態とその遷移方法を定義したり，プログラミングをする必要があり手作業によるテスト準備に時間がかかっていた．本研究では一連の作業を容易にするためのシステムを実装した．システムは，(1)GUI より GUI 情報を取得して，ユーザ操作，補完操作を作成する OdbEditor，(2)ユーザ操作を組合わせてテストケースを作成する ScriptEditor から構成される．このシステムは Windows95 以降のすべてのバージョンで動作する．開発環境は VC++6.0 を使用した．

次にシステムの各機能や利用方法について述べる．

4.2. OdbEditor

OdbEditor(図 7)は，GUI をキャプチャして GUI 情報を取得する機能をはじめ，以下の 3 つの主な機能を持つ．

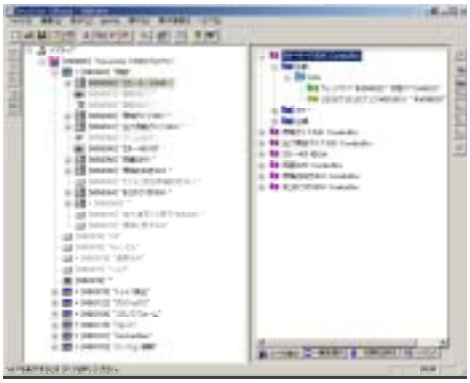

図 7 OdbEditor

・GUI情報の取得
自動実行したいWindows GUIをポイントしてGUI情報をキャプチャする．OdbEditorでは取得したGUI情報をWindowsのウィンドウと同じ単位，構造で，WinObjというオブジェクトで管理する．WinObjにはキャプチャと同時に一意のIDを付加する．ID，テキスト,クラス名などは編集することもできる．ウィンドウがサブダイアログを持つ場合は，テスト対象のすべてのウィンドウをキャプチャする必要がある．

・ユーザ操作の作成
テストしたいWinObjをユーザ操作を管理しているビューにドラッグ＆ドロップすることでユーザ操作が作成される．ドラッグ＆ドロップすると，WinObjの属性値であるクラスから適切なコマンドを選択して作成すると同時にWinObjのオーナー関係をGUI情報から求めて適切なpre-stateを作成する

・補完操作の作成
補完の関係を記述したい2つのWinObjを選択して，補完操作の作成を指示する．次に，コマンドのターゲットとなるWinObjに対してのコマンドとpost-stateのターゲットとなるWinObjの状態を指定することにより，補完操作を作成する．タブのプロパティシートを表示させる操作とツリーコントロールのアイテムを選択したときに動的に表示されるコントロールについてはGUI情報の取得時に自動的に補完操作を作成する．

4.3. ScriptEditor

ScriptEditor(図 8)は,OdbEditorで作成したGUI情報，ユーザ操作，補完操作が記述された情報を参照してテストスクリプトを編集する機能をはじめ，以下の機能を持つ．

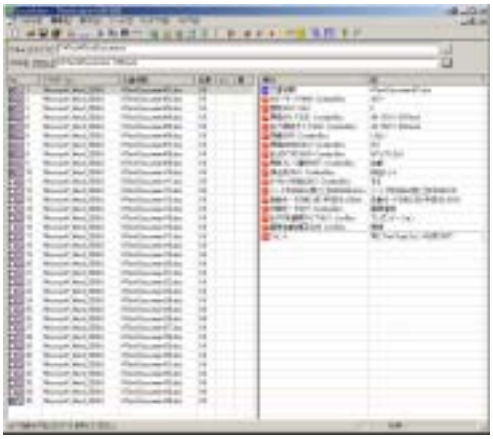

図 8 ScriptEditor

・テストスクリプトの編集
テストスクリプトは，複数のテストケースから成り，テストケースはユーザ操作と値の組の集合である．補完操作は自動実行時に自動的に補完シーケンスとして実行されるのでテストケースには含まれない．テストスクリプトは実行したいユーザ操作だけを選択するのでテストスクリプトが非常にシンプルになり，テストスクリプトの再利用性を高めることができる．

・テストケースの自動実行

実行したいテストケースをスクリプトから選択する．テストケースにはチェックボックスがついており，テストしたいテストケースだけを選択できるようになっている．自動実行をおこなうと，選択したテストケースを順番に実行していく．ユーザ操作と値の組を実行するときに，システム内部で補完操作探索をおこない補完シーケンスが構成される．補完操作が行われて状態が変化したら，その都度補完操作探索をおこないダイナミックに補完シーケンスを再構築する．実行中にエラーとなった場合には，そのテストケースを中止し次のテストケースの実行に移る．エラーとなった情報は実行終了時にログとして報告される．エラー状態の画面を画像ファイルに残しておくこともできる．次節でプリンタドライバ(図 9)を使ったシステムの実行例を説明する．

5. テスト実行例

5.1. プリンタドライバ

以下にシステムの実行例として弊社のプリンタDocuColor1250のプリンタドライバを使用する(図 9)．テストケースとしては表 3を考える．テストケースの設定項目で出力用紙サイズと原稿の向きは図 9のダイアログの中に存在する．なお，テストケースのアプリケーションWord2002日本語版のユーザ操作と補完操作は別途作成してあり,ScriptEditorで指定する(表 4)．

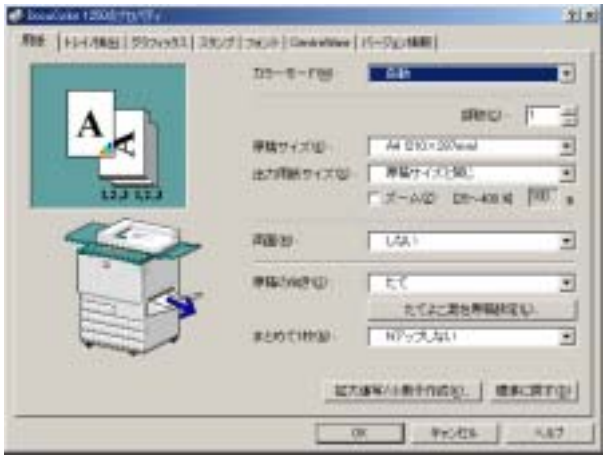

図 9 DocuColor1250

表 3 プリンタドライバのテストケース

テストケース	
アプリケーション	Word2000日本語版
テスト文書	TestDocument01.doc
出力用紙サイズ	A3
原稿の向き	たて

表 4 Word2002 日本語版のユーザ操作/補完操作

Word2002日本語版	
ユーザ操作	文書を開く
補完操作	アプリケーションを起動する
	文書を閉じる
	ファイルダイアログを開く
	ファイルダイアログを閉じる
	印刷ダイアログを開く
	プリンタプロパティを開く
	プリンタプロパティを閉じる
	印刷(印刷ダイアログを閉じる)

まずOdbEditorにより，プリンタドライバのGUI情報をキャプチャする.取得したGUI情報のWinObjツリーとWindowsの階層構造および状態より補完操作を自動的に作成する(図10).

GUI情報の取得が終了したら,テストしたいユーザ操作を作成する．ユーザ操作として実行したい操作は以下のGUIである．

- 出力用紙サイズ
- 印刷の向き

用紙タブシートの中にあるWinObj”印刷の向き”コンボボックスをユーザ操作にドラッグ＆ドロップする．この時，システムはWinObj”原稿の向き”が以下であることを利用して自動的にユーザ操作を作成する．

- クラスがComboBoxである
- アイテムに”たて”および”よこ”を持つ
- 親ウィンドウが用紙タブプロパティシート

同様に，印刷の向きのユーザ操作を作成する(図 11)．

ユーザ操作の作成が終了したら，次に補完操作の作成をおこなう．タブプロパティシートを前面に出す補完操作は作成されているので，プリンタドライバのメインウィンドウを閉じる補完操作を作成する．補完操作の作成は専用のダイアログにおいて指定する．関係する２つのWinObjを指定して，補完操作の作成を指定すると補完操作が作成される.post-stateは図 12で指定したWinObjのIDと状態により確定されるが，pre-stateは,この時OKボタンと親子関係にある用紙タブプロパティシートより,自動的に作成される(図 13)

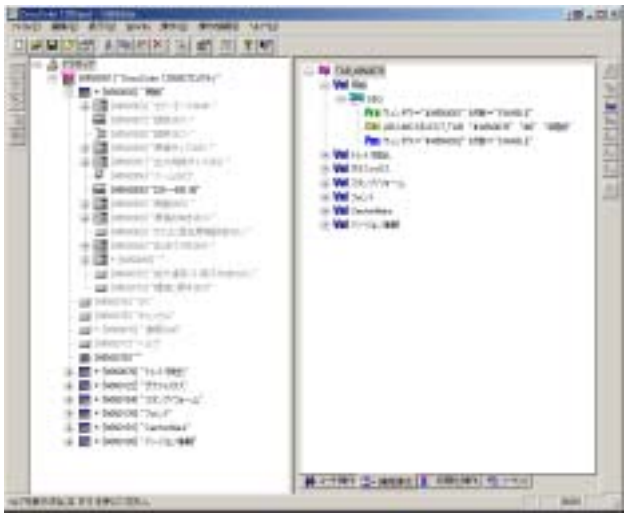

図10 自動的に作成された補完操作

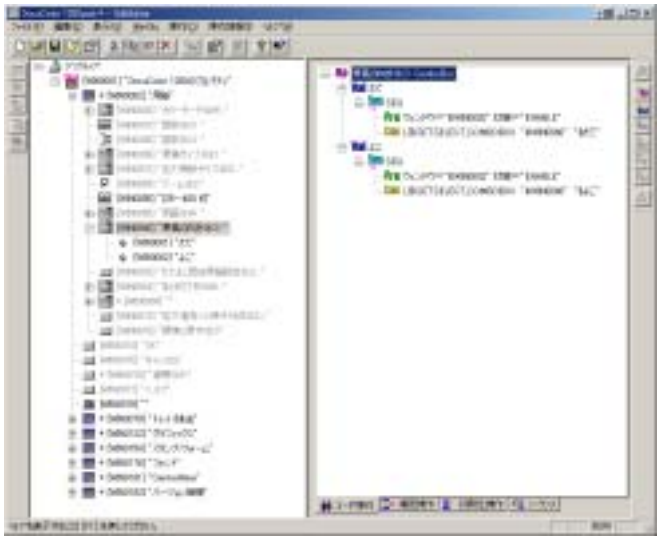

図 11 ユーザ操作の作成

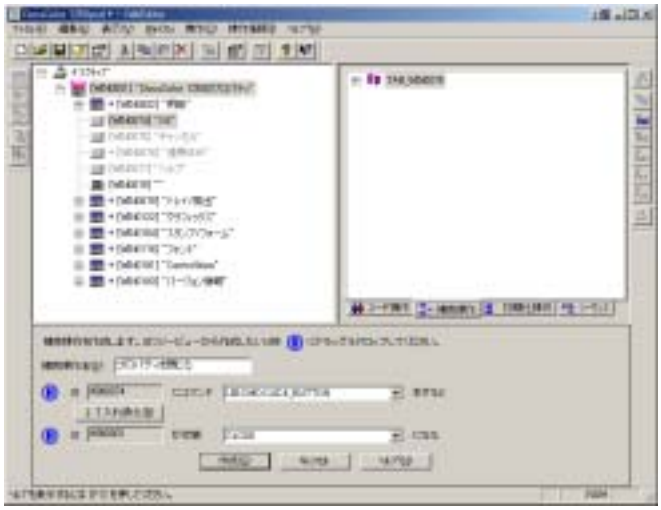

図 12 プロパティを閉じる補完操作作成

図 13 作成された補完操作

5.2. テストケースの作成と実行

ScriptEditor において，テストスクリプトを新規作成してテストケースを作成する(図 14).テストケースに実行のチェックをして，実行を開始する．開始すると，現在の GUI の状態を調べて，テストケースで指示されたユーザ操作を実行するための補完シーケンスを構築しダイナミックに操作を補完しながら実行していく．実行中に，図 15のようなコンファーマが出ると，どのような対処を実行するのかを問い合わせるコンファーマ登録ダイアログを出す（図 16）．ここでは，テストドキュメントが変更されて保存されないように[いいえ]のボタンをクリックするコマンドを登録して，実行を再開する．自動実行が終了すると，どのような補完操作が実行されたのかを示す実行ログを表示して終了する．実行ログよりテストケースで指定したユーザ操作に対して，どのような操作が補完されたかがわかる(図 17).

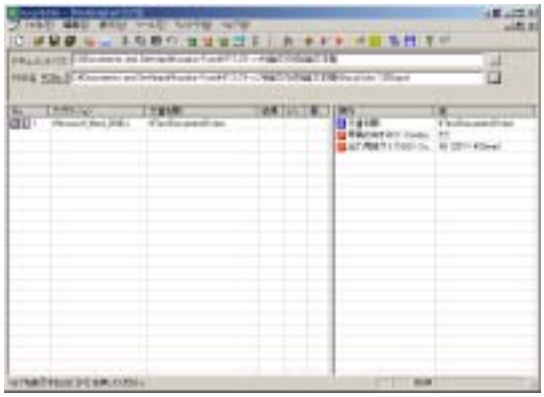

図 14テストケース

図 15 コンファーマ

図 16 コンファーマの登録ダイアログ

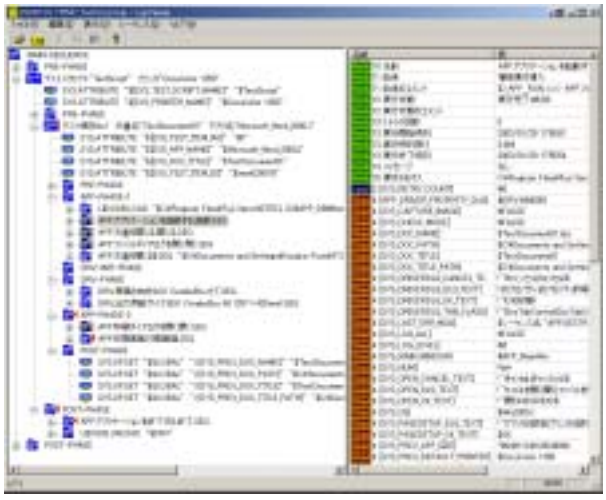

図 17 実行ログ

6. 考察

6.1. ダイナミック補完によるテスト実行の課題

ダイナミックに操作を補完するテストの自動実行の効果については 1 節で述べた．一方，検討すべき課題として補完シーケンスが複数の経路を持つ場合の網羅性という点がある．補完シーケンスはシステムが自動的に補完操作探索をおこない，実行するので補完シーケンス経路の網羅性に対して保証できない．補完シーケンスの経路が複数あった場合にはどちらを選択するかについて統計的解析に基づく決定アルゴリズムの追加が必要かもしれない．

6.2. 自動テストシステムの応用と課題

自動テストシステムとしての応用と課題に関しては以下のようなものを検討中である．

- Webアプリへの応用
  考慮できる状態やシーケンスを増やして，一般的なWebアプリに応用する．
- 機能の網羅性
  現在のシステムではユーザ操作を評価対象の重要機能と考えた場合に機能の網羅性などを保証するテストをおこなう場合には，スクリプト作成に時間がかかる．ユーザ操作を評価対象の重要機能として作成し，組合せ網羅性を保証するツールとの連携によって手動ではおよそ不可能な複雑な自動実行を効率よくおこなう．
- GUIの仕様変更に対するGUI情報の同期
  現状の機能においてもGUIの仕様変更を検知する機能はあるが，さらに効率良くテストをおこなうためには，仕様変更されたGUIとシステム内の自動化するための情報を自動的に同期する仕組みが不可欠である．

7. おわりに

GUI をともなうソフトウェアの自動テストにおいて，ダイナミックに操作を補完しながら実行するシステムを作成した．これによって，GUI 仕様変更時のデータの作り直しを減らし，何回も繰り返しおこなわれるテストを容易に実施できるようになった．

しかし現在は，テストの実行としての適用事例が自社内のアプリケーションやプリンタドライバに限られているため，今後は汎用性を高め，他の開発などに適用し有効性を十分に確認していこうと考えている．最後に弊社のプリンタのシステムテストのデータ作成に要した工数(表 5)およびプリンタのシステムテストにおける自動実行システムの導入率の推移(図 18)を示す．

表 5 データ作成工数

ユーザ操作数	72
補完操作数	36
テストケース数	128
データ作成工数(単位:人日)	1.5

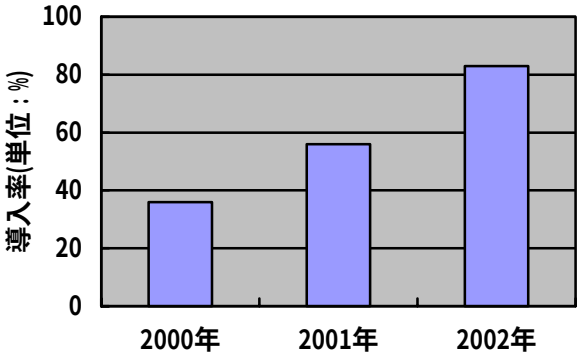

図 18 弊社のシステムテストへの導入率

文　献

[1] 小谷　千里，新日本製鐵株式会社，特開平9-223042，公開特許公報(A)，1998.

[2] C. Kaner, J. Falk, H. Q. Nguyen “基本から学ぶソフトウェアテスト”, pp.158，日経 BP 社, 2001.