ソフトウェアのテスト効率と精度を向上させるGUI画面の自動操作技術

Automatic GUI Operation Technology for Efficient and Accurate Software Testing

平井　潤　　関根　智　　川野　晋一郎
■ HIRAI Jun　■ SEKINE Satoshi　■ KAWANO Shin-ichiro

GUI（Graphical User Interface）を備えたソフトウェアを開発する際の品質管理では，テスト工程作業の効率化と精度の向上が重要である。GUIのテストを自動化する技術やツールは従来からあるが，テストシナリオの作成に高度な知識を必要とし，テスト対象となるソフトウェアのアーキテクチャにも制限があった。

そこで東芝ソリューション（株）は，GUI画面の自動操作技術を用いたテスト技術を開発した。これは，画像探索技術を用いることにより，リッチクライアントを含む幅広いアーキテクチャによるソフトウェアのテストに適用できる可能性が高い。このテスト技術は，既にGUIテストツール GUIPilot$_{TM}$として実装され，当社のソフトウェア開発現場で活用されている。

In the quality management of software with a graphical user interface (GUI), there is a need to improve the efficiency and accuracy of the testing process. However, the conventional software testing tools require advanced knowledge in order to make the test scenarios, and the architecture of the software that can be tested is also subject to limitations.

To overcome these problems, Toshiba Solutions Corporation has developed a software testing technology that performs automatic GUI operation. It has high applicability to the testing of software with various GUI architectures including rich clients, due to the adoption of an image searching technique. This software testing technology, implemented as the GUIPilot$_{TM}$ GUI testing tool, is already being applied to software development in our company.

1　まえがき

ソフトウェア開発の品質管理では，テストの重要性は極めて高い。テストには，例えば，機能が仕様どおりに動作するかを検査する機能テスト，仕様どおりの処理性能が実現できているかを検査する性能テストなどがある。これらのテストを手動で行う際に，多数のテスト項目に対するGUI操作や，多人数での端末の一斉操作が必要となる。これは効率が低いうえに，作業者の苦痛を伴う。また，操作の再現性に乏しいためテストの精度が低下する。

このため，テストの効率と精度を向上させるには，ツールによるテストの自動化が不可欠である。GUIのテストを自動化する技術やツールは従来からあるが，テストシナリオの作成に高度な知識を必要とし，テスト対象となるソフトウェアのアーキテクチャにも制限があるという課題がある。

そこで東芝ソリューション（株）は，画像探索技術を用いて，GUI画面を人間に代わって操作する新しいGUIテスト技術を開発した。これをテストツール GUIPilot$_{TM}$として実装することで，上記課題の解決に取り組んでいる。

ここでは，この新技術の特長と，ポイントとなるシナリオ生成技術について述べ，適用事例を示す。

2　GUI自動操作技術の概要

テストの自動化を実現するために，GUIをAPI（Application Program Interface）として用いる技術[1]を適用した。

GUIPilot$_{TM}$によるGUI自動操作の概要を**図1**に示す。GUIPilot$_{TM}$は，GUI操作手順があらかじめ記述されたシナリオを読み取り，その手順に従ってOS（基本ソフトウェア）を介

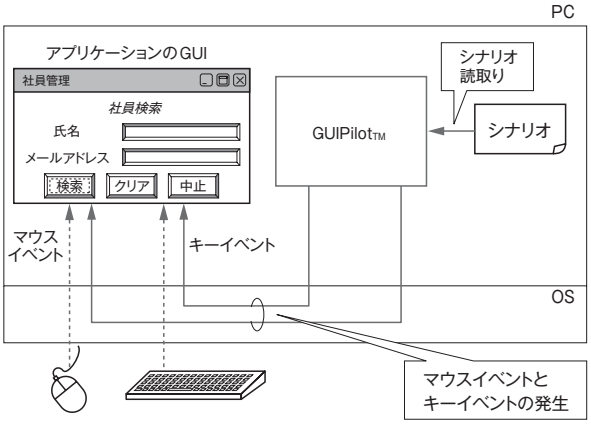

図1. GUIPilot$_{TM}$によるGUI自動操作の概要 — GUIPilot$_{TM}$は，シナリオに従いマウスイベントやキーイベントを発生させてGUIを自動操作する。
Overview of automatic GUI operation by GUIPilot$_{TM}$

してマウスイベント(注1)やキーイベント(注2)を自動的に発生させることにより，アプリケーションのGUIを操作する。マウスとキーボードから出ている破線の矢印は，手動操作の場合のイベントの流れで，ここではGUIPilot$_{TM}$によってこの手動操作を自動的に模擬している。

3　画像探索方式の特長

GUIアプリケーションのシステム全体を対象に行うシステムテスト（総合テスト）を実施する際の手法を**表1**に示す。各方式には（A）～（D）の記号を便宜的に付与しており，GUIPilot$_{TM}$は，画像探索方式（D）を実装したものである。画像探索方式（D）は，ほかの方式と比較しシナリオ作成に要求される知識が少なく，また多種の対象アーキテクチャに汎用的に適用できるとういう優位性を持っている。

ここでは，画像探索方式（D）の特長と優位性について述べる。

表1．ソフトウェアのシステムテストの実現方式
Methodologies of software system testing

システムテストの方式			記　号
手　動			（A）
自　動	プロトコルエミュレーション方式		（B）
	GUI自動操作方式	オブジェクト指定方式	（C）
		画像探索方式(*)	（D）

＊：テストツール GUIPilot$_{TM}$に実装

3.1　GUI自動操作方式（C），（D）とプロトコルエミュレーション方式（B）

システムテストの中でも特に，性能テストを実施するためにシステムに負荷を発生させる方式の観点から，表1の各方式を比較する（**図2**）。

性能テストでは，サーバに大量の処理要求を送出して負荷を与えるために，多人数を動員してクライアントのGUIを手動で操作する方式（A）か，又は負荷発生ツールを用いて自動化する方式（B）～（D）を採る。負荷発生ツールには，プロトコルエミュレーション方式（B）とGUI自動操作方式（C），（D）とがある。

プロトコルエミュレーション方式（B）は，テストツールが多数のクライアントの代わりに，サーバへ処理要求を送出する方式である。

一方，GUI自動操作方式（C），（D）は，クライアントのGUIを人間の代わりに自動操作することによって，サーバに対して負荷を発生させる方式である。

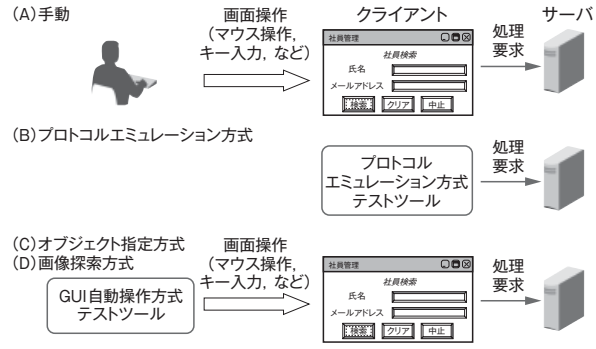

図2．性能テストにおける負荷発生方式の比較 ― 性能テストでは，サーバに大量の処理要求を送出して負荷を与えるために，多人数を動員してクライアントのGUIを手動で操作する方式（A），又は負荷発生ツールを用いて自動化する方式（B）～（D）を採る。

Comparison of load generation methodologies in performance testing

GUI自動操作方式とプロトコルエミュレーション方式はともに長所と短所があるが，小規模な性能テストではGUI自動操作方式に優位点が多い。各々の長所と短所について以下に述べる。

3.1.1　シナリオ作成に求められる知識　両方式ともに，テストの手順をシナリオに記述する必要があるが，そのために要求される知識には大きな差がある。

プロトコルエミュレーション方式（B）では，どの処理要求をどの順序でサーバに送出するかをシナリオに記述する。近年のWebシステムをはじめとするサーバでは，処理要求や応答の一部分が動的に変化することが多い。そのため，シナリオを記述するためには，処理要求及び応答の構造や，クライアントの処理要求の一部が動的に変化するロジックに関しての知識が必要である。したがって，シナリオの記述が難しくミスを誘発しやすい。

一方，GUI自動操作方式（C），（D）では，動的な部分を含む処理要求の生成は，実際のクライアントそのものを動作させることで実現できるため，ユーザーはGUIの操作に関する知識さえあればシナリオを作成できる，という特長がある。

3.1.2　適用対象となるソフトウェアの制約　プロトコルエミュレーション方式（B）では，処理要求の通信方式が限定され，適用対象となるソフトウェアへの制約が多い。一方，GUI自動操作方式（C），（D）では，適用対象となるソフトウェアは通信方式の点で制約を受けない。

3.1.3　性能テストの規模　プロトコルエミュレーション方式（B）は，少ないCPU能力で大量の負荷を発生できる利点がある。一方，GUI自動操作方式（C），（D）は，一つのOS上で一つのアプリケーションしか動作しないケースも多いため大量の負荷を発生しにくい場合があり，大規模な性能テストに

（注1）　コンピュータの動作を発生するためのマウスの位置やマウスボタンの状態の変化情報。
（注2）　コンピュータの動作を発生するためのキーボードからの入力操作に関する情報。

は向いていない。

3.2　画像探索方式（D）とオブジェクト指定方式（C）

ここでは，画像探索方式（D）の特長を，オブジェクト指定方式（C）と比較して述べる。ここでの比較は，3.1節で述べた性能テストだけでなく，機能テストでの活用も想定している。

GUIは通常，ウィジェットやコントロールとも呼ばれている，プッシュボタンやラジオボタンなどのGUI部品の集合体として構成されている。

オブジェクト指定方式（C）は，各々のGUI部品のオブジェクトとしてのID（Identification）を明らかにして，どのGUI部品オブジェクトに対してどのようなマウスイベントを発行すべきかをシナリオに記述する方式である。これらのGUI部品を管理する仕組みや構造は，Microsoft® .NET® Framework(注3)やJava®(注4)のSwing(注5)のように，GUIを実現する環境により異なっている。したがって，オブジェクト指定方式（C）は，各々のGUI部品の管理構造に対応する必要があり，テストツールの開発に多大なコストがかかるうえ，新しいGUI部品が開発されるたびにそれに対応する必要がある。このように適用できるソフトウェアアーキテクチャが限定されるのが課題である。

一方，画像探索方式（D）は，GUI部品の管理構造には依存しない。この方式のツール GUIPilot$_{TM}$におけるシナリオの概念を**表2**に示す。シナリオでは，操作対象画像情報，マウス操作，及びキー入力に関する情報が一つのセットとなっており，GUIPilot$_{TM}$はそれを上から順に実行して行く。

シナリオの一つ目の手順を実行するようすを**図3**に示す。GUIPilot$_{TM}$は，まずシナリオから操作対象画像情報を読み取り，パソコン（PC）のデスクトップ上に表示されているアプリケーションのGUI画像の中から，ピクセル単位で一致する箇所を比較探索する。GUI画像の中でクリアボタンの箇所が発見されると，GUIPilot$_{TM}$はその座標へマウスを移動させ，シナリオのマウス操作に従って左クリックのイベントを発生させ，GUIを自動操作する。

このように，画像探索方式（D）のGUIPilot$_{TM}$は，表示されているGUIの画像情報に基づいて動作するために，GUIを実現する仕組みや管理構造には依存しない。したがって，オブジェクト指定方式（C）よりも多くの種類のGUIアーキテクチャに汎用的に適用できる可能性が高く，例えば種々のリッチクライアントへの適用も期待できる。

表2．シナリオの概念
Concept of scenario

手順	操作対象画像情報	マウス操作	キー入力
1	クリア	左クリック	−
2	レス	左クリック	“Hirai.Jun”
3	検索	左クリック	−

（注3）　Microsoft，.NET Frameworkは，米国Microsoft Corporationの米国及びその他の国における登録商標。
（注4）　Javaは，米国Sun Microsystems,Inc.の米国及びその他の国における商標又は登録商標。
（注5）　プログラミング言語JavaのGUIを構築するためのツールキット。

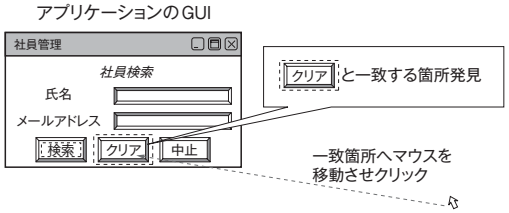

図3．画像探索によるGUI自動操作 — デスクトップ上の画像から，シナリオの操作対象画像情報であるクリアボタンの画像と一致する箇所を発見し，そこにマウスを移動させクリックを行う。
Automatic GUI operation using image searching technique

4　シナリオ生成技術の開発

画像探索方式（D）には，3章で述べたような優位性があるが，効率的なシナリオ作成の手法がないのが課題であった。そこで当社は，ユーザーがアプリケーションを操作する手順をモニタしてシナリオを生成する技術を開発した。

GUIPilot$_{TM}$は，ユーザーがマウスやキーボードを手動で操作して発生したマウスイベントやキーイベントの情報をOSから取得し，デスクトップ上に表示されている画像情報と合成して，シナリオを生成する（**図4**）。

シナリオ生成の過程について**図5**を用いて述べる。GUIPilot$_{TM}$は，ユーザーがマウスを左クリックしたときの座標をOSから取得し，デスクトップ上のアプリケーションGUIの画像情

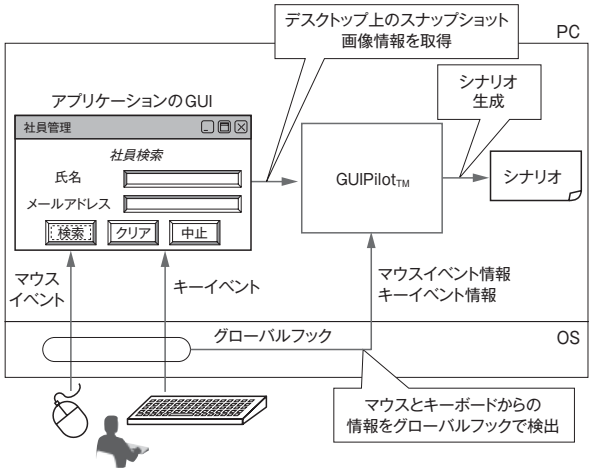

図4．シナリオ生成の概要 — ユーザーがGUIを手動で操作する際に，デスクトップの画像とマウス・キーイベントを取得して，シナリオを生成する。
Overview of scenario generation

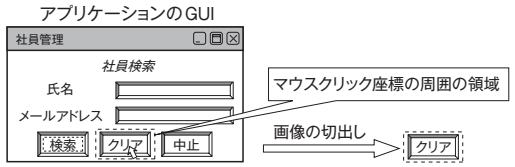

図5. シナリオ生成 — デスクトップ上の画像から，ユーザーがマウスでクリックした座標周囲の領域を切り出し，操作対象画像情報として保存する。
Scenario generation

報から，その座標の周囲の領域を切り出す。切り出した画像をシナリオの操作対象画像情報として保存するとともに，左ボタンをクリックしたという情報をマウス操作として保存する。このようにして，ユーザーがGUIを操作する手順をシナリオとして生成する。

5　GUIPilot$_{TM}$の適用事例

当社は，ソフトウェアの開発現場でGUIPilot$_{TM}$の適用を進めており，その事例について以下に述べる。

5.1　繰返し安定動作の検証テスト

信頼性を求められるアプリケーションにおいて，多数回の操作でも安定動作することを検証するテストにGUIPilot$_{TM}$を適用した事例について述べる。

この事例では，あるPCアプリケーションのインストールとアンインストールによるバージョンアップ作業を5,000回繰り返し行い，異常が発生しないことを検証する必要があった。

GUIPilot$_{TM}$を利用した場合と，手動で実施することを想定した場合での作業時間の比較を**表3**に示す。

GUIPilot$_{TM}$を利用した場合には，ツールの習得，シナリオの作成，及び自動テストの準備作業に8時間程度を要したが，その後は5台のPCを並列に動作させ，昼夜を通して1週間程度で自動化したテストを完了できた。

同等のテストを手動で実施することを想定すると，単純作業を833時間行う必要があり，現実的に実施が困難であった。また，結果を短期間で得たい場合には交代勤務で夜間も作業しなければならないという不都合も想定された。

このように，GUIPilot$_{TM}$の適用により，ソフトウェアの品質を確保するためのテストを短期間に効率良く実施することができた。

表3. 手動で実施時とGUIPilot$_{TM}$を利用時の作業量の比較
Comparison of workloads using manual operation and GUIPilot$_{TM}$

実施方法	作業時間（h）
手動で実施	833
GUIPilot$_{TM}$を利用	8

5.2　クライアント同時運転による安定動作の検証テスト

150台のPCでクライアントアプリケーションを同時に操作した状態での，クライアントサーバシステムの安定動作を検証するテストを実施した。

手動で実施すると，少なくとも30人のオペレーターを動員して，各々5台のPCに対して一斉に操作を行う必要がある。一方，GUIPilot$_{TM}$を利用すると，1人分の労力でこのテストを実施でき，大幅な省力化と，GUI操作の精度向上によるテストの品質向上が実現できた。

6　あとがき

ソフトウェアのテストの効率化を実現するGUI画面の自動操作技術，及びテストツール GUIPilot$_{TM}$の技術的特長と事例について述べた。GUIPilot$_{TM}$は従来技術と比べ、シナリオ作成が容易で，多種のアーキテクチャのソフトウェアへ適用できると思われる。

当社は生産性，品質，及び保守性の向上を目的としたシステム開発基盤CommonStyle$_{TM}$の整備を行っている。CommonStyle$_{TM}$は，多くのフレームワーク，コンポーネント，ツール，及びガイドから構成されており，GUIPilot$_{TM}$もその一つとして全社的に活用し効果を上げている。

今後も，開発現場からの要望や提案を受け，GUIPilot$_{TM}$機能の向上と使いやすさの改善を継続し，ソフトウェアのテストにおける効率と品質の向上に貢献したい。

文　献

(1)　山本格也．GUIをAPIとして用いるプログラミング法．情報処理学会論文誌：プログラミング．**39**，SIG1（PRO1），1998，p.26－31.

平井　　潤　HIRAI Jun
東芝ソリューション（株）IT技術研究所 研究開発部主任。ソフトウェアの品質管理技術，及び性能管理技術の研究・開発に従事。
Toshiba Solutions Corp.

関根　　智　SEKINE Satoshi
東芝ソリューション（株）IT技術研究所 研究開発部主任研究員。ソフトウェアの品質管理技術，及び性能管理技術の研究・開発に従事。
Toshiba Solutions Corp.

川野　晋一郎　KAWANO Shin-ichiro
東芝ソリューション（株）IT技術研究所 研究開発部。ソフトウェアの品質管理技術，及び性能管理技術の研究・開発に従事。
Toshiba Solutions Corp.

一般論文