Vol.2012-SE-178 No.9
2012/11/1

GUI アプリケーションのテスト駆動開発の一手法

石井良亮[†1]　　岸知二[†2]

近年，テスト駆動開発という手法が注目されているが，GUI アプリケーションに対する適用には課題がある．本研究では先行研究で用いられた低忠実度プロトタイプの考えを用い，GUI アプリケーションのテスト駆動開発をする一手法を提案する．

A Method of Test-Driven Development of GUI Applications

YOSHIAKI ISHII[†1]　TOMOJI KISHI[†2]

Nowadays, technique of test-driven development is being noticed. However, being unrefined, the method has problems about applying to GUI applications. In this research, we propose a method of test-driven development of GUI applications with the ideas of low-fidelity prototype.

1. はじめに

近年，様々なプログラム開発手法が考案されている．中でも，信頼性の確立やバグの低減などの目的のため，テスト駆動開発(Test Driven Development 以下 TDD と略す)が注目されている．しかし，TDD は現在洗練中の技術であり，グラフィカルユーザーインターフェース（Graphical User Interface 以下 GUI と略す）への適用などの課題 [1]があり，実際の開発に用いることが難しい ．

先行研究では，GUI アプリケーションへの TDD の適用のためアプリケーションがどのように動くかという例示であるストーリーボードを用いてユーザーとアプリケーション間のインタラクションを明示化した．そこから低忠実度プロトタイプと名づけた簡易的な GUI を用いてインタラクションを記録することでテストケースを生成し，GUI アプリケーションのテスト駆動開発を可能にするアプローチを取った．しかし，先行研究では GUI に関する表現力が弱く，行えるテストが限定的であった．

そこで，本研究ではストーリーボードに代わるより表現力の高いモデルを用いることでこの課題を改善することを目標とする．これは，アプリケーションの挙動のモデルからモデルベースドテスト（Model Based Test 以下 MBT と略す）のアーキテクチャの様にテストケースを生成することを提案する．

本論では，2 章で TDD と GUI に関する研究背景及び同じ研究目的の先行研究について，3 章でモデルベースドテストについて，4 章で本研究のアプローチについて，5 章で例題について，6 章で TDD と MBT の関係について現時点での考察を述べる．

†1　†2　早稲田大学
Waseda University

2. 研究背景

2.1 TDD について[1][2][3]

TDD は Kent Beck によって提唱されたアジャイルなプログラム開発手法の一種で，プログラムに必要な機能について最初にテストを作成し，テストにパスする実装を行った後にコードを洗練させるという短いサイクルを繰り返す手法である．図 1 に TDD のアーキテクチャを示す．

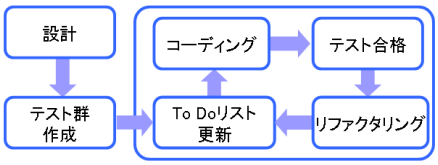

図 1　TDD のアーキテクチャ

TDD を用いる利点は多数ある．開発者と顧客の間のコミュニケーションを促進させること，プログラマーの信頼性を向上させること，ソフトウェアの品質を向上させること，生産性を低下させることなくバグ密度を低下させることなどである．このような利点があるため TDD は注目されているが，洗練中の技術であり GUI の自動テストや分散オブジェクトの自動ユニットテストなどがし辛い（ダラク・エニスの挑戦[1]）などの欠点がある．

2.2 本研究における GUI のテスト

本研究では，図 2 のような構造を持つ GUI アプリケーションを想定して研究を行うが，その中でも範囲を絞って研究を行う．

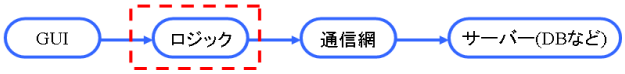

図 2　想定するアプリケーションの仕組みと本研究の範囲

サーバーサイドのテストはデータベース内のコンテキストに依存するため，テストファーストにテストを作成することが難しい[4]. そのため，今回は扱わない．次に，GUI 部分はロジック部と入出力データをやり取りするための部分であり，複雑なロジックを必要としないため，これも範囲外とする．したがって，本研究では図 2 のような GUI アプリケーションのロジック部をテスト対象とし，GUI アプリケーションのテストと呼ぶものとする．

よって，本研究での大目標は GUI アプリケーションの開発に TDD を適用できる様にすることである．

2.3 解決すべき課題

手動で GUI をテストすることは単体テストよりも時間がかかり難しいとされているため，GUI アプリケーションをテストするためのテスト方法が望ましいとされている．これは一般的なテキスト UI のアプリケーションのインスタンスに比べ, GUI は多くの設定を必要とするからである．例えば，マウスやキーボードから入力されるユーザーとのインタラクションの設定などが挙げられる[5]. そのようなユーザーの振る舞いをシミュレートする GUI のテスト手法の一つとしてキャプチャリプレイツール（capture replay tool 以下 CRT と略す）を用いたものがある[6].

CRT とはユーザーと GUI とのインタラクションを記録し, GUI 上で再生をするためのツールである. これにより，GUI のテストを半自動的に行うことが出来る．しかし，このツールを用いて GUI のテストを行う際には GUI が実際に実装されていないとインタラクションが記録できないので，実行が不可能である．そのため，GUI が実装される前の環境では CRT による GUI アプリケーションのテストは出来ず，テストファーストなテスト設計が出来ないため，TDD のアーキテクチャに反してしまう．また，付随する問題として CRT を用いたテストはデータ入力に用いられる GUI エレメントの位置情報に依存するが，GUI エレメントはユーザーにとって使いやすい，あるいは操作ミスが起こらないように GUI エレメントのレイアウトや大きさを変えることがある．これも解決しなくてはならない問題である．

2.4 先行研究[7]

本節では GUI の TDD に関する先行研究について述べる．

2.3 節で述べた様に，GUI の TDD は難しいとされているが，この根本的な難しさはウィジェット内のメソッドにテストコードで容易にアクセスできないためである. そこで，ユーザーストーリーから UI の様々な状態の画像と“ホットゾーン”と名前を付けた状態間の遷移の原因となるプロトタイプの特定の部分によって構成されている低忠実度プロトタイプを考案した．この低忠実度プロトタイプは GUI を実装する前に CRT に必要なインタラクションを記録するための GUI のスタブとして機能する．これを用いることで CRT を用いた GUI のテストが可能となる．

先行研究ではまず，ユーザーがどのようにアプリケーションを操作するかの例である手書きのストーリーボードを作成する．図 3 は先行研究の例題で用いた計算機のストーリーボードである．

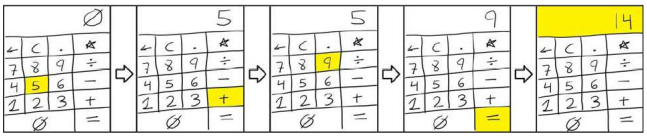

図 3　先行研究におけるストーリーボード[7]

このストーリーボードを用いて，必要最低限の GUI を実装する低忠実度プロトタイプを作成する．この低忠実度プロトタイプは GUI の見た目だけの実装であり，実際のロジック部は何も作られてはいない．図 4 はこの例題における低忠実度プロトタイプの例を示す．この例では各ボタンがホットゾーンに相当する．

図 4　低忠実度プロトタイプの例[7]

低忠実度プロトタイプの各ホットゾーンには自動的に ID が割り振られ，GUI 内の対応するウィジェットに使われていることを保証する．

これより，実際の GUI を実装する前に，プロトタイプ内でインタラクションを記録することが可能になる．この時に記録されたインタラクションをテストケースとして用いることで GUI の TDD を可能にした．図 5 は先行研究における低忠実度プロトタイプを用いて作成したテストケースの例である．

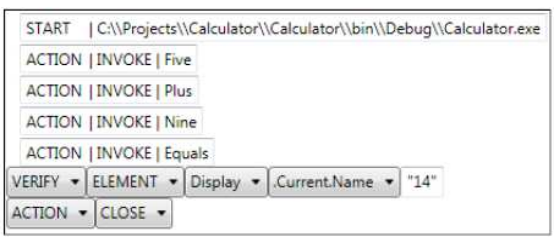

図 5　低忠実度プロトタイプを用いて作成したテストケース[7]

先行研究では主に以下の 2 つのことを行ったと考えられる．

1)GUI アプリケーションの挙動のモデル化

先行研究ではユーザーストーリーを集めるためにストーリーボードを利用した．GUI のテストを記録するために CRT の使用が望ましい．しかし，そのためには GUI の実装を完全にしなければならず，テスト駆動のアーキテクチャに沿わない．そこで，ストーリーボードを用いてテストに必要な部分の挙動をモデル化した．これはユーザーとアプリケーションのインタラクションを GUI の実装前に明示化し，アプリケーションの入出力データの組を例示化したものであると考えられる．

よって，GUI アプリケーションに対する入出力データ，及びそれに関連するホットゾーンを明らかにすることが出来た．

2)ストーリーボードとテストケースの対応

テスト駆動を行うためにはアプリケーションの要件に対応するテストケースが必要となる．特に，GUI アプリケーションの場合に必要とされるテストケースはユーザーの挙動と期待される出力結果の組である．

先行研究では 1)で考えたモデルを表現したプロトタイプ上のエレメントとテストケース上のエレメントをそれぞれに対応する ID で関連付けを行った．これにより，モデル上のエレメントとテストケース上のエレメントの同一視が可能となったため，視点の違う 2 つのモデルが出来た．ストーリーボードでは GUI の挙動を表し，テストケース上では GUI のロジック部のテストを表すことができた．

以上より，GUI のテストを考える際に GUI 部分とロジック部分を分けて考えることにより GUI 部分のみを先に作成し，CRT によって GUI のテストを記録することを可能にした．そして，CRT によって記録されたインタラクションをテストケースにすることで GUI の TDD を可能にした．

3. MBT について[8]

MBT とはテスト設計モデルを用いてテストケースを設計する技術の総称である．機能網羅テストや状態遷移テストなどが例として挙げられる．図 6 に MBT のアーキテクチャを示す．

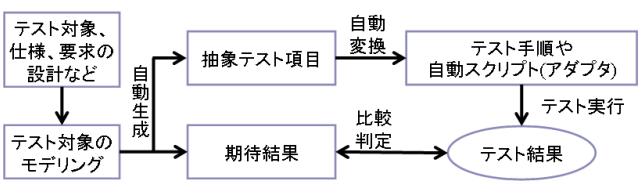

図 6　MBT のアーキテクチャ[8]

テスト設計モデルとは，テスト対象を何らかの形でモデル化したものであり，モデルの種類はどのようなものでも良い．また，MBT では対象の何をテストしたいかによって使用するモデルの種類が変わる．例えば，システム全体の状態遷移をテストしたいときには状態遷移モデルを用いる，システム内の要素の組み合わせを見たいときには直交配列表などの組み合わせモデルを用いる，などの様に使い分けを行う．

4. 研究アプローチ

4.1 先行研究の問題点と研究目標

先行研究では，モデルの表現できる範囲としてストーリーボード上で表せたのは GUI エレメントをクリックするという動作だけであった．そのため，先行研究の問題点としてクリックだけで使用できる GUI エレメントにしか手法を適用できなかった点が上げられる．これは， GUI アプリケーションのモデルとしてはストーリーボードでは表現力が乏しく，テストできるものが限られていたためである．そこで，先行研究で実装が出来なかった GUI エレメントの実装を可能にすることを本研究の目標とする．

本研究では GUI アプリケーションをテストする際は単純にボタンをクリックするだけでなく，ラジオボタンの選択やテキストボックスへのテキストの入力についてもテストが必要であると考えた．そのため，本研究ではテキスト入力やボタン選択などの GUI エレメントに対する操作の表現をモデルに取り入れることでモデルの表現力を増やし，先行研究の問題点を改善することを考えた．具体的には，本研究で想定している GUI アプリケーションのテストはクライアントサイドの GUI のロジックのテストのため，クライアントサイドの GUI の実装に広く用いられている JavaSwing での GUI エレメントの実装を行う．

本研究では先行研究と同様に，GUI の挙動面のモデルとロジック面のテストケースを分離することを考えている．中でも，ロジック部分を TDD で開発することを目指しているため，GUI を実装する前にインクリメンタルなテストケースを用意する必要がある．そのため，ロジック面のテストケースはロジックの入力データと期待される出力結果の組み合わせとなる．しかし，GUI の実装前では入出力データがどのような形式であるかを指定できないため，GUI アプリケーションの挙動モデルを先に作成し，そこからロジックの入出力データの形式を抽出することを考えた．これによりインクリメンタルにテストケースを作成できるため，TDD のアーキテクチャに当てはめることが出来ると考えられる．

図 7 に本研究の全体像を示す．

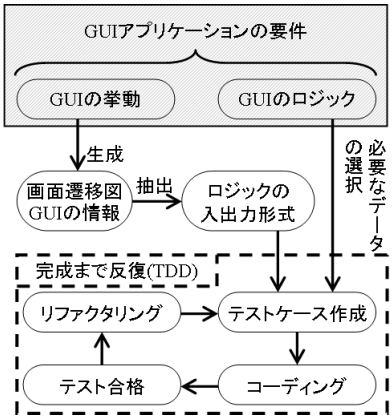

図 7　本研究の全体像

まず，作成したい GUI アプリケーションの要件から GUI がどのようなロジックによって動くかを画面遷移図にする．そこから各状態間を遷移する際のロジックに必要な入出力の形式を抽出する．そこに，GUI アプリケーションの要件からロジックのテストをするために必要なデータを選択し，当てはめることで，テストケースが生成できる．これにより，GUI の実装前にテストケースが生成できるので TDD のアーキテクチャに従い GUI アプリケーションを開発することができると考えている．

4.2 テストモデル要件

本研究では，先行研究と同じく GUI における GUI の挙動面とロジック面を分けて考えるため， GUI アプリケーションの挙動面のモデル化とロジック面のテストケースの生成を考えなくてはならない

この中でも，ロジック面についてはテスト駆動を適用する部分なので，入力データと出力データを指定されたテストケースを複数用いてテストファーストかつインクリメンタルにテストを行う．

一方，2.2 節で述べたように本研究では GUI アプリケーションの挙動についてはテスト駆動を適用しない．GUI アプリケーションの挙動モデルは MBT のアーキテクチャに従い，テストケースを生成するために用いる．MBT ではモデルに表現されていない部分はテストが行われないため，一度に全体像を表すモデルが必要である．

1)　GUI アプリケーションのロジック面

GUI アプリケーションの挙動を示したモデルより，各ロジックに必要な入出力データの形式が得られるため，それに合うようにテストケースを作成することでインクリメンタルに GUI アプリケーションのテストが可能になる．テストケースは GUI エレメントの入力するデータの組と出力結果の組み合わせである．ただし，先行研究の例にあるように入力順序で結果が変わるアプリケーションもあるため，入力順序も表す必要がある．

2)　GUI アプリケーションの挙動について

GUI アプリケーションの挙動のモデルでは，システムの全体像を表すことが必要である．そして，ロジック面のテストに必要な入出力データの形式を確定するため，ロジックの前後に GUI アプリケーションの画面上にどのような GUI エレメントが存在するかを表す拡張をする必要がある．そのため，ロジックによる GUI アプリケーションの状態や画面の変化を表せる画面遷移図を拡張して用いる．

まず，本研究における画面遷移図はステートマシーン図の拡張である．GUI アプリケーションの画面及びロジックの前後の状態を，ステートマシーン図における状態として扱ったものに GUI エレメントの情報の拡張を行う．これより，本研究におけるロジックはモデル上の状態間の遷移の際に処理が行われる．これは，画面遷移のないロジックについても，ロジックが行われるタイミングをモデル上で状態遷移の間に限定し，全てのロジックについて入出力データの形式を抽出するためである．そして，各画面間の遷移はきっかけとなるアクションや遷移するための制限をトリガーシグニチャとガード条件として表現する．ロジック面はこのモデルとは別にテストケースを作成するため，GUI アプリケーションの挙動のモデルには表現されない．

次に，画面遷移図に行う拡張は各画面に存在する GUI エレメントを表現するものである．ロジック面のテストケースの入出力情報を抽出するために，各 GUI エレメントの種類と名前が分かることが必要である．また，入出力データの形式を指定するために GUI アプリケーションの挙動モデルを先に作成する．

図 8 はこれらの要件を満たす今回の GUI アプリケーションの挙動モデルのメタモデルである．

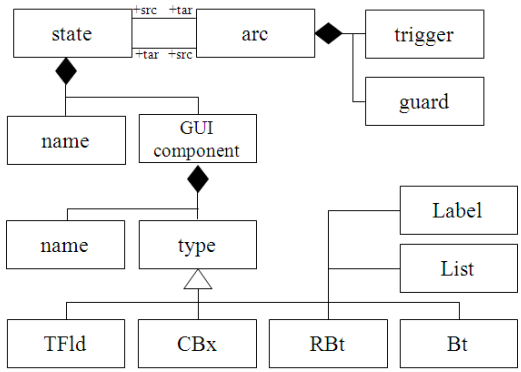

図 8　GUI アプリケーションの挙動モデルメタモデル

このメタモデルでは state が GUI アプリケーションの状態（画面）を示し，arc が状態間の遷移を表す．arc によって状態が遷移している間にロジックが処理されると想定しているので，arc には trigger と guard によって遷移の制限が

可能となっている．また，state にはロジックの入出力として必要な GUI エレメントの情報を持たせることが出来る．このエレメントの属性は種類と名前しかないが，エレメントの種類が決まればそれに対する操作と入力に必要なデータの形式がわかるため，それらの属性は持たせない．

5. 例題

5.1 例題シナリオ

今回の例では身長・体重・性別・年代を入力し，BMI と個人の肥満度をアバウトに表示するアプリケーションを取り扱う．このアプリケーションはテキストボックスによる数値の入力とラジオボタンによる性別の選択，リストボックスによる年代の選択，ボタンをクリックすることによる画面の遷移を含む．また，画面遷移の間に BMI を計算するロジックと BMI から肥満度を選択するロジックが含まれる．

このアプリケーションはクライアントサイド上で単独で稼動できるデスクトップアプリケーションを想定していて，入力にはマウスとキーボードを用いる．

このアプリケーションのストーリーボードを図 9 に示す．

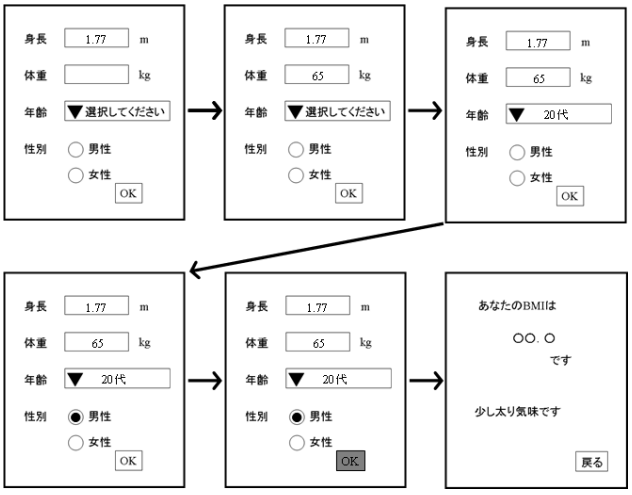

図 9　具体例のストーリーボード

今回は例示のため，ストーリーボードを掲載した．しかし，実際に本手法を用いる際には GUI アプリケーションの挙動モデルからテストケースに必要な入出力の形式を抽出するため，実際のエレメントをテストの設計段階では意識する必要がない．そのため，実際に本手法を用いる時にはストーリーボードは作成しなくてもよい．

5.2 例題モデル

具体例のアプリケーションの主な画面は情報の入力画面と出力画面である．今回は GUI の挙動をモデル化する際に input ページと output ページと名前を付けた．このアプリケーションの全体の挙動を図 10 に示す．

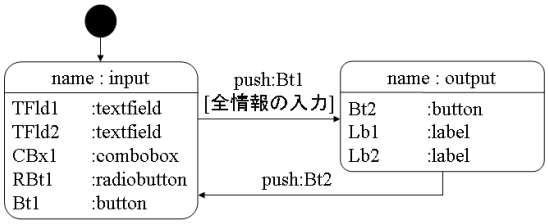

図 10　GUI アプリケーションの挙動のモデル

このテストモデルより，アプリケーションの挙動と遷移に必要な GUI エレメントの情報が分かる．そのため，GUI エレメントとそれに入力する情報を組み合わせることで，テストを行うことが可能となる．

具体的な input ページから output ページへの遷移へのテストケースは表 1 の様になると考えている．

表 1　本研究のテストケース入力例

input				output	
エレメント名	入力順序	値		エレメント名	値
TFld1				Lb1	
TFld2				Lb2	
CBx1				Bt2	
RBt1					
Bt1					

表 1 の input の入力順序，値及び output の値の欄にテストケースで用いたい値と，それを入力する順序を入力する．これで，input ページから output ページに遷移する際の入力と出力の組み合わせができる．これを用いてロジックについてのテストをインクリメンタルに作成することが可能となる．先行研究と比較し，入力データとして数値データを扱えるため，テキストフィールドやコンボボックスなどが扱えると考えている．

5.3 評価について

本研究の評価はまだ検討中の段階である．先行研究との比較，妥当性や GUI エレメントのテストとして必要十分であるかなどが考えられるが現在検討中である．具体的な方針としては簡単な 2〜3 個のアプリケーションを提案手法を用いて開発を行ってみて，その結果を現在調査中の評価項目で評価をする予定である．

6. 考察

6.1 TDD と MBT の原則

TDD では原則としてテストを念頭に入れて作成されていないコードを TDD に移行させることは非常に困難である．そのため，TDD ではすべてのコードを書く前にテストを作成する必要があるので，最終的にはアプリケーション

全体についてのテストを作成する必要がある．つまり，テスト作成⇒コーディングを繰り返し，アプリケーション全体を作成するインクリメンタルな手法である．

一方, MBT はテスト対象を表したモデルからテストケースを作成する．また，モデルに表現されていない仕様はテストケースに反映されないため, MBT に用いられるテストモデルはテスト対象を全体的に表す必要がある．

ただし，これは一つのモデルでテスト対象全体を表現する手法である．

6.2 TDD と MBT の関係

以上のことから，TDD と MBT はテスト作成の方法は対照的であるが，いずれもテスト対象全体についてのテストを作成する必要がある．これより，TDD と MBT を組み合わせることでいくつかの利点があるのではないかと考えられる．

7. 参考文献

[1] Kent Beck・長瀬嘉秀，「テスト駆動開発入門」，ピアソンエデュケーション，2003

[2] 川端光義 倉貫義人 兒玉督司 長瀬嘉秀，「バグがないプログラムの作り方」，翔泳社，2004

[3] Jonathan Rasmusson 西村直人 角谷信太郎,「アジャイルサムライー達人開発者への道」，オーム社発刊，2011

[4] Scott W.Ambler，”test-driven development of relational database”，IEEE Software ，2007

[5] MIrko Stocker ，「InfoQ “JRuby での JavaGUI テスト”」，http://www.infoq.com/jp/news/2008/11/gui-testing-jruby，2008

[6] Vladimir Entin, Mathias Winder, Bo Zhang, Stephan Christmann，” Combining Model-Based and Capture-Replay Testing Techniques of Graphical User Interfaces”，2011 Fourth International Conference on Software Testing, Verification and Validation Workshops，2011

[7] Theodore D. Hellmann・Ali Hosseini Khayat・Frank Maurer，” Supporting Test-Driven Development of Graphical User Interfaces Using Agile Intertype Design”， Third International Conference on Software Testing, Verification, and Validation Workshops，2010

[8] Nishi Yasuharu，”モデルベースドテスト入門”，ソフトウェアシンポジウム 2007 東京，2007