# GUI プログラミング課題自動採点方式の検討

立石 良生† 井上 潮‡

†東京電機大学 工学研究科 情報通信工学専攻 〒120-8551 東京都足立区千住旭町 5 番
‡東京電機大学 工学部 情報通信工学科 〒120-8551 東京都足立区千住旭町 5 番
E-mail: †18kmc10@ms.dendai.ac.jp, ‡inoueu@mail.dendai.ac.jp

**あらまし** 近年，プログラミング教育が重要視されている．オブジェクト指向のプログラミングを習得するために GUI のプログラミング演習を行うが，GUI の採点は描画されたアプリケーションのレイアウトを確認するため，非常に手間がかかる．既存の GUI 自動テスト手法は特別な環境が必要で，作成にスキルと時間を要する．そこで，レイアウトの採点作業を自動化するとともに，外部ファイルを用いることで採点用プログラムを容易に作成できるシステムを実現した．実際の授業の演習課題を用いて学生のプログラムで評価した結果，自動採点用プログラムの作成時間を短縮でき，教員が手動で採点するのと同じ精度で学生のプログラムを自動採点できることを確認した．

**キーワード** オブジェクト指向，プログラミング，演習課題, GUI，自動テスト

## 1. はじめに

近年，人工知能（AI）や IoT などのテクノロジーが急速に進化していることからプログラミング教育が重要視されている．文部科学省は，2020 年度より小学校でプログラミング教育を必修化することを明示しており，その後中学校と高等学校でも必修化が実施されることになっている[1]．このことからプログラミング言語に興味を示す学生およびプログラミング授業を行う教育機関は増加することが考えられる．

初心者向けのプログラミング言語として Java が用いられることが多い．Java の学習ではコマンドラインインタフェース（CLI）のプログラムを作成する他に，オブジェクト指向プログラミングを習得するためにグラフィカルユーザインタフェース（GUI）のプログラムを作成する．プログラミング言語の習得は実際にソースコードを記述することが非常に重要なため，プログラミングの授業において毎回数個の簡単なプログラムを作成する課題が課されることが多い．しかし，数百人分の学生のプログラミング課題を毎回手作業で採点するのは非常に手間がかかり,教員の負担が大きい．特に，GUI プログラミング課題を採点する場合は描画された GUI のレイアウト確認やマウスでの操作，文字の入力などが必要なためより多くの時間を必要とする上に採点ミスも発生しやすくなる問題がある．そこで本研究では学生が作成した GUI プログラミング課題を自動で採点するシステムを開発し，プログラミング教育の効率化を目的とする．

本論文は以下のような構成になっている．第 2 章で関連研究を紹介し，従来の手法の利点と問題点を述べる．第 3 章では前提条件として開発環境や使用するライブラリ，自動テスト方式についての説明をする．第 4 章では静的テストについて我々が提案する手法を述べる．第 5 章で実際のプログラミング課題を用いて自動採点の実験を行う．第 6 章で手動採点と比較して自動採点システムの評価をし，第 7 章で全体のまとめと今後の課題を述べる．

## 2. 関連研究

GUI アプリケーションの自動テストに関する研究は多く行われている．代表的な手法としてスクリプトベース，モデルベース，キャプチャ/リプレイ方式がある[2]．スクリプトベースは GUI のレイアウト，イベントのテストを記述し，プログラム上で動作させる手法である[3]．モデルベースは，テストケースを自動で生成するモデルを作成しテストを行う手法である[4]．キャプチャ/リプレイ方式は事前に GUI を動作させ，イベント処理と結果の画面を記録する．テストを実行する際に記録した GUI 操作を自動的に実行し，実行結果が記録されたものと比較する手法である[5]．この手法は GUI 操作のみで自動テストを行うことができるが，ウィンドウに描画する処理が必要なため複数の GUI をテストするのに時間がかかる．これらの手法を用いて開発された自動テストツールについて紹介する．

井上[6]は，学生が作成した演習課題の GUI プログラム を自動でテストするシステムを開発した．これはスクリプトベースで開発されており，JavaFX で記述された GUI アプリケーションを自動で採点することができる．テストスクリプトは JavaFX のみで記述されているため，簡単に作成することができる．しかし，1 つのテストスクリプトを作成するのに 1 時間程度かかることから，多種類の GUI をテストするのには多くの時間を必要とすることが課題として挙げられていた．

Nguyen ら[7]は，「GUITAR」と呼ばれる GUI アプリケーション自動テスト用ツールを開発した．これはモデルベースのフレームワークとなっており，GUI テスト対象のコンポーネントやイベントを容易に拡張する

ことができる.これにより,従来の手法と比較して様々な GUI アプリケーションの自動テストに柔軟に対応できるようになった.しかし,手動での作業や誤検出,予期せぬエラーなどが発生する問題が挙げられていた.

このように GUI を自動でテストするシステムは多く開発されているが，事前準備に時間を必要とすることや,採点精度が悪いなどの問題がある.本研究では,自動テスト，採点するためのスクリプト作成時間を短縮し，手作業での採点と同じ精度で採点ができる手法について検討する.

## 3. 前提条件

### 3.1. JavaFX

Java の GUI ライブラリとして代表的なものは Swing と JavaFX である．Swing は AWT の拡張ライブラリとしてよく用いられていたが，Java8 からは JavaFX が標準 GUI ライブラリとして導入されている[8].本研究では，JavaFX で作成された GUI プログラムを対象とする.

JavaFX には多くのコンポーネント(部品)，レイアウトペインが存在し，様々な GUI アプリケーションを容易に作成することができる．代表的なレイアウトペインとコンポーネントをそれぞれ表 1，表 2 に示す.

表 1　代表的なレイアウトペイン

| 名称 | 説明 |
|---|---|
| **HBox** | コンポーネントを水平に配置 |
| **VBox** | コンポーネントを垂直に配置 |
| **BorderPane** | 上、左、右、下、中央の各位置に配置 |
| **GridPane** | 行と列の柔軟なグリッド内に配置 |
| **FlowPane** | 境界で折り返されるフローに配置 |

表 2　代表的なコンポーネント

| 名称 | 説明 |
|---|---|
| **Label** | 文字を出力するラベル |
| **TextField** | 文字を入力できる領域 |
| **Button** | イベントを登録できるボタン |
| **RadioButton** | 選択肢の中から 1 つ選ぶボタン |
| **ComboBox** | 選択肢を提供するリスト |
| **CheckBox** | 選択肢を提供するボックス |

JavaFX での GUI 作成は以下の手順で行う.

① ボタン，ラベルなどのコンポーネントを作成
② コンポーネントにイベントを登録
③ コンポーネントをレイアウトペインに配置
④ レイアウトペインをシーンに入れる

JavaFX の GUI アプリケーションの構成の例を図 1 に示す.

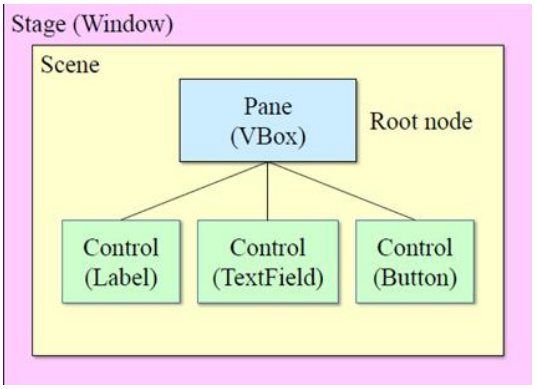


図 1　GUI アプリケーションの構成の例

### 3.2. 対象のプログラミング課題

本研究では東京電機大学情報通信工学科の学部生を対象とした授業「オブジェクト指向プログラミングおよび演習」のプログラミング課題を対象とする．この授業は全部で 14 回行われ，演習課題が毎回 2 問程度出題される．演習課題は授業で習った基本的な事項を身に付けることを目的としているため，作成する GUI は簡単な機能を搭載したものである．実際に出題された課題の例を紹介する.

例：DenominationApp.java･･･入力された金額に対する金種を計算する GUI アプリケーション

① レイアウトに VBox を使用し，HBox を 7 つ配置する
② HBox1 に Label1，TextField，Label2 を，HBox2 に Button1，Button2 を，HBox3 に Label3 を，HBox4～7 に Label を 2 つずつ配置する
③ Button の下の Label(金額エラー)を非表示にし，中央揃えに設定する

この GUI アプリケーションのレイアウト画面を図 2 に示す.テキストフィールドに金額を入力し,「金種計算」ボタンを押すと下に各紙幣の枚数が表示される.「リセット」ボタンを押すと，紙幣の枚数が全て 0 になる．テキストフィールドに自然数以外を入力しボタンを押すと，下に「金額エラー」と赤文字で表示される.このような GUI アプリケーションを作成する課題が毎回出題されている.


図 2　DenominationApp.java

### 3.3. 静的テスト/動的テスト

GUI のテストは大きく分けて「静的テスト」と「動的テスト」に分けられる[9]．静的テストは，GUI のボタンやラベルなどの部品が正しく配置されているかを検証するテストである．動的テストは，ボタン押下などのイベント処理が正しく行われているか検証するテストである．GUI のテストでは予期せぬエラーを防ぐため，最初に静的テストを行い，レイアウトが正しいことが確認できた場合のみ動的テストを行う．本稿では静的テストの手法に焦点を当てて説明していく．

## 4. 提案手法

### 4.1. システム構成

プログラミングの授業で演習課題を課す場合，教員は事前に模範解答のプログラムを作成し，動作確認を行う．この模範解答を用いて静的テストを行い，自動採点するシステムについて説明する．学生が作成した GUI の採点は以下の手順で行う．

① 教員は模範解答と学生の解答のプログラムを用意する
② これらの GUI からレイアウト情報をそれぞれ取得し，XML 形式でファイルに保存
③ 2 つの XML ファイルを比較して採点する
④ 採点結果を出力する

この手順で採点を行うことで学生の課題を自動で採点することができ，他の GUI を採点する際も新たに自動テスト用スクリプトを作成する必要がなくなるので，採点者の負担を大幅に減らすことができる．このシステムの構成図を図 3 に示す．

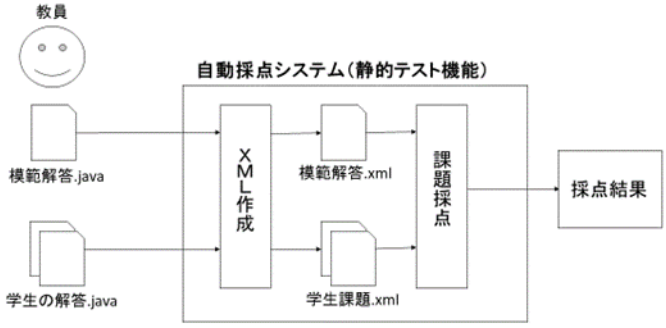


図 3　自動採点システム構成図

### 4.2. 模範解答.xml の作成

採点用の XML ファイル（模範解答.xml）を自動生成するために，XML ファイル自動生成プログラムを用いる．XML ファイル作成プログラムの処理フローを図 4 に示す．

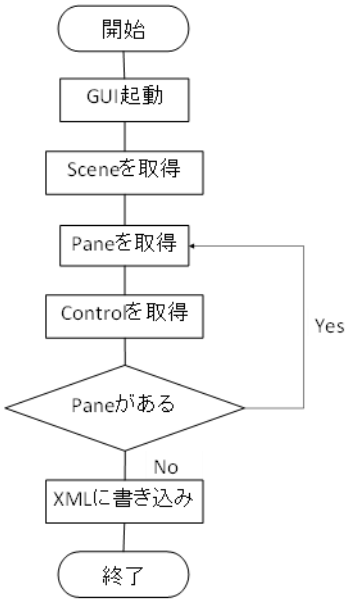


図 4　XML ファイル自動生成プログラムの処理フロー

最初に教員は採点したい課題の模範解答を用意する．この模範解答から GUI のシーンを取得し，シーンに入っているレイアウトペインを取得し，レイアウトペインに入っているコンポーネント情報を取得し，これらを XML 形式で保存する．具体的には，コンポーネントとレイアウトペインを XML のタグとして，コンポーネントに設定した文字をタグのテキストデータとして保存する．また，同じ名前のタグを区別するため，それぞれのタグに ID を与え，属性として保存する．ID は上から順に 0, 1, 2, …を与える．レイアウトペインで

は，全体の位置を決める Alignment のデータも属性として保存する．

### 4.3. 学生の課題の採点方法

次に模範解答の XML ファイルを用いて学生の解答を自動採点する手法について説明する．4.2.と同様の手法で,学生が作成した GUI プログラムを用いて XML ファイルを作成する．その後，学生の XML ファイルと模範解答の XML ファイルからレイアウト情報を読み込み比較していくことで採点を行う．XML ファイルを 1 行ずつ読み込み，タグ，属性が一致しているか確認する．テキストデータは採点に考慮しないことが多いため，採点対象から除外する．点数は 1 つのタグが模範解答の XML と一致した場合，1 点を与える．

## 5. 実験

### 5.1. 実験方法

実際に授業で扱った演習課題を用いて実験を行う．4 章で示した手法で教員が作成した模範解答から XML ファイルを自動生成し，学生の解答と比較し採点結果を表示させる．実験で用いる演習課題は 3.2.で紹介した金種計算アプリケーションで行う．この GUI にはレイアウトペインが 8 個，コンポーネントが 14 個あるので，合計点は 22 点となる．

### 5.2. 実験結果

金種計算アプリケーションの XML ファイルは図 5 のようになる．図 5 を見ると，図 2 で示したアプリケーションのレイアウト情報が正しく配置されていることがわかる．また，ID を与えたことにより，図 5 のような同じ名前のタグが複数ある場合でも区別しやすい．図 5 の模範解答の XML ファイルを用いて,学生の XML ファイルと比較して静的テストを行うと，結果は図 6 のようにコマンドラインに表示されるようになっている．全て正解しているので，項目ごとに「OK」と表示されている．間違っていた場合は「NG」と表示され，1 点減点される．

```
<Scene ID="0">
<VBox ID="1" Pos="CENTER">
    <HBox ID="2" Pos="CENTER">
        <Label ID="3">金額</Label>
        <TextField ID="4">0</TextField>
        <Label ID="5">千円</Label>
    </HBox>
    <HBox ID="6" Pos="CENTER">
        <Button ID="7">金種計算</Button>
        <Button ID="8">リセット</Button>
    </HBox>
    <HBox ID="9" Pos="CENTER">
        <Label ID="10">金額エラー</Label>
    </HBox>
    <HBox ID="11" Pos="CENTER">
        <Label ID="12">10000円札</Label>
        <Label ID="13">0</Label>
    </HBox>
    <HBox ID="14" Pos="CENTER">
        <Label ID="15">5000円札</Label>
        <Label ID="16">0</Label>
    </HBox>
    <HBox ID="17" Pos="CENTER">
        <Label ID="18">2000円札</Label>
        <Label ID="19">0</Label>
    </HBox>
    <HBox ID="20" Pos="CENTER">
        <Label ID="21">1000円札</Label>
        <Label ID="22">0</Label>
    </HBox>
</VBox>
</Scene>
```

図 5　自動生成された XML
(DenominationApp.java)

```
<終了> XMLTester
HBox : OK
Label : OK
Label : OK
HBox : OK
Label : OK
Label : OK
HBox : OK
Label : OK
Label : OK
HBox : OK
Label : OK
Label : OK
点数 : 22/22
```

図 6　学生の課題の静的テスト結果(一部)
(DenominationApp.java)

## 6. システムの評価

### 6.1. 評価方法

5 章で示した静的テストの自動採点システムを用いて，122 名の学生のプログラムで自動採点を行う．授業の演習課題の中からレイアウトの作成を演習としているプログラミング課題の一覧を表 3 に示す．

表 3　レイアウト作成課題

| 番号 | 名称 | 内容 |
|---|---|---|
| ① | CheckSIDApp.java | 学籍番号の正誤判定 |
| ② | AddTaxApp.java | 税込み価格を計算 |
| ③ | CountUpDownApp.java | ボタンで設定した数値が上下する |
| ④ | DenominationApp.java | 入力金額の金種を計算 |
| ⑤ | TicketCalculator.java | チケット価格計算 |

これら 5 つのうち，図 2 で示した DenominationApp.java を除く 4 つの GUI のレイアウト画面を図 7，図 8，図 9，図 10 に示す．

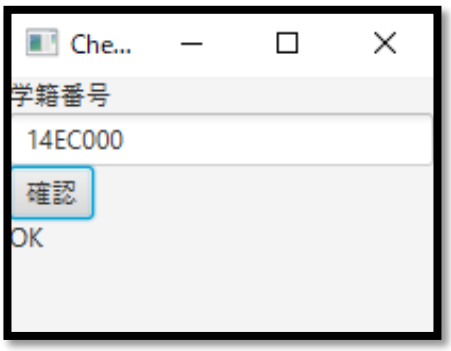


図 7　CheckSIDApp.java

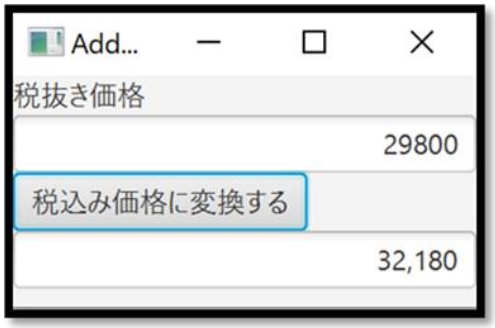


図 8　AddTaxApp.java


図 9　CountUpDownApp.java


図 10　TicketCalculator.java

### 6.2. 採点精度の評価

これらの課題を用いて，提案手法と手作業で採点したときの精度を比較して評価を行う．ここでの採点基準はレイアウトペインとコンポーネントが全て正しく配置されていた場合に正解とする．これら 2 つの手法で学生の課題を採点した結果のうち，AddTaxApp.java の採点結果において提案手法と手作業での採点（手動採点）の正解と不正解の数を混同行列で示したものを表 4 に示す．なお，履修者 122 名のうち，未提出者が 16 人，コンパイルエラーとなったのが 5 人だったので，採点対象者は 101 名である．

表 4　学生の課題の採点結果
(AddTaxApp.java)

| 101 名 | | 提案手法 | |
|---|---|---|---|
| | | 正解 | 不正解 |
| 手動採点 | 正解 | 95 | 0 |
| | 不正解 | 0 | 6 |

表 4 を見ると，手動で採点した結果が「正解」だったものは提案手法でも全て「正解」となり，「不正解」

だったものは全て「不正解」となっている．この評価方法を残りの 4 つの演習課題で行った結果，全ての演習課題が提案手法と手動採点の結果が一致した．この結果から全ての演習課題の自動採点が手作業での採点と同じ精度を出していることがわかる．これは演習課題において使用するレイアウトペインコンポーネントの配置を指示していたので，正解者の GUI が模範解答と完全一致していたと考えられる．しかし，実際の採点では必ずしも模範解答と完全に一致している必要はない．演習課題で指示していない部分も採点対象となってしまう可能性もある．教員が事前に採点する項目を選択し，選択した項目だけで点数をつけることができるよう改善する機能を追加する必要がある．

### 6.3. 採点時間の評価

次に採点時間の評価を行う．ここでは表 3 で示した 5 つの課題の採点時間を計測し，既存の自動採点システムと比較する．比較に用いる既存システムは，2 章で紹介した，井上が開発した自動採点システムを用いる．提出された課題を 1 つずつ自動採点し，処理時間を計測する．この処理時間を提出者全員分足し合わせたものを「採点時間」とする．この方法を用いて既存手法と提案手法で自動採点し，計測した処理時間を表 5 に示す．

表 5　学生の GUI 採点時間

| 課題番号 | 採点<br>対象者[人] | 採点時間 [s] | |
|---|---|---|---|
| | | 既存手法 | 提案手法 |
| ① | 98 | 36.7 | 38.6 |
| ② | 101 | 37.6 | 42.5 |
| ③ | 91 | 33.0 | 38.1 |
| ④ | 99 | 38.1 | 42.1 |
| ⑤ | 97 | 42.1 | 46.1 |

表 5 を見ると，全ての演習課題で提案手法の採点時間が既存手法より遅いことがわかる．提案手法は XML ファイルを読み込むため，既存手法より処理に時間がかかっていることが考えられる．しかし，約 100 人分の採点時間の差が 2 秒～5 秒なので，ほとんど問題はないと思われる．また，既存手法では教員が演習課題ごとに自動テスト用プログラム(スクリプト)を約 1 時間かけて作成していたが，提案手法では模範解答のみで全ての課題採点が行えるため，5 つの課題採点を約 5 時間短縮することができたと言える．

## 7. まとめ

GUI プログラミング課題の静的テスト方式において，部品やレイアウトを描画せずに XML ファイルに格納するプログラムを作成したことにより，様々な GUI アプリケーションで簡単にレイアウト，部品を取得することができた．XML ファイルを用いて学生の課題を自動でテストした結果，手作業で採点するのと同じ精度を出すことができた．採点にかかる時間は既存の手法よりやや劣ったものの，模範解答の GUI のみで自動採点ができることから，自動テスト用のプログラムを作成する時間を短縮することができた．しかし，学生の GUI アプリケーション採点する際に，採点対象の項目を決めることができないなどの問題点が挙げられた．また，レイアウトはボタン押下などのイベントにより動的に変化することも多いので，動的なレイアウト画面の評価方法についても検討していく必要がある．本研究では静的テストに焦点を当てたが，実際の演習課題にはイベント処理の追加などの課題も出題されており，動的テストを自動化する手法についても検討していかなければならない．今後の課題として採点項目の選択機能や動的なレイアウトの変化を評価する手法，イベント処理などの動的テストの自動化が挙げられる．

## 参　考　文　献


[1] 小学校プログラミング教育の手引(第二版)，平成 30 年 11 月，文部科学省．

[2] Rui Carvalho, “A Comparative Study of GUI Testing Aproaches”, Faculdade de Engenharia da Universidade do Porto, 2016.

[3] Snyder, J., Edwards, S.H., Pérez-Quiñones, “LIFT: taking GUI unit testing to newheights”, Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, pp. 643–648, 2011.

[4] X. Yuan, M. B. Cohen, and A. M. Memon, “GUI interaction testing: Incorporating event context,” Softw. Eng. IEEE Trans., vol. 37, no. 4, pp. 559–574, 2011.

[5] E. Börjesson and R. Feldt, “Automated system testing using visual GUI testing tools: A comparative study in industry,” in Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on, 2012, pp. 350–359.

[6] Usio Inoue, “GUI Testing for Introductory Object-Oriented Programming Exercises”, 5th International Conference on Computational Science/ Intelligence & Applied Informatics, pp. 1-13, 2018.

[7] Bao N. Nguyen, Bryan Robbins, Ishan Banerjee, Atif Memon, “GUITAR: an innovative tool for automated testing of GUI-driven software”, Autom Softw Eng, pp. 65-105, 2014.

[8] JavaFX の概要, https://www.oracle.com/technetwork/jp/java/javafx/overview/index.html.

[9] 内藤広志，斉藤隆，水谷泰治，“GUI プログラミング課題の自動採点方式について”，情報処理学会研究報告，p81-88, 2008.