卒 業 研 究 報 告 書

題　目　　J2EEを用いたWEBアプリケーションに対する
工数見積りツールの開発

指導教員　　水野　修　准教授

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　08122056

氏　　名　　椋代　凜

平成24年2月15日提出

J2EE を用いた WEB アプリケーションに対する工数見積りツールの開発

平成 24 年 2 月 15 日　　08122056　椋代　凜

概　要

本研究では，J2EE を用いた WEB アプリケーションのテスト工数を見積るテスト工数見積りツールの開発を行う．見積り方法には，積分法と専門家による判定の二つの方法を用いる．WEB アプリケーション実行時の WEB ブラウザの画面上に存在する入力系統（GUI）の情報をソースコードから取得し，それぞれに予め設定したテスト工数を掛け合わせた上で合計し，それを対象の WEB アプリケーションのテスト工数とする．本開発ツールの適用実験には，J2EE を用いた WEB アプリケーションの設計サンプルとして Oracle から提供されている「The Java Petstore」を対象として用いる．実験の結果，The Java Petstore の WEB ブラウザ画面全てを対象としたテスト工数を見積ることができた．しかし，各画面毎の遷移情報と入力系統の情報の取得が困難なことから，機能テストやシナリオテストなどの WEB アプリケーションの使用ユーザの要求を考慮したテスト作業は補えないことがわかった．よって，本開発ツールは WEB アプリケーションに含まれる入力系統の種類や属性，個数といった情報と WEB アプリケーション全体のテスト工数を提供することで，同等のサイズの WEB アプリケーションの開発，及び入力系統のテストには有用であるが，このツールだけでテスト工数の見積り作業の全てを補うことは困難であるということがわかった．今後の課題としては，WEB アプリケーション実行時の各画面毎の情報の取得と，画面遷移情報の取得，及び GUI 部の改良がある．

目　次

1. 緒言

WEB アプリケーションは，特定のソフトウェアをインストールする必要もなく簡単な WEB ブラウザの操作だけで使用できることから，インターネット上や組織内の情報システムで頻繁に使用されている．WEB アプリケーションには様々なアプリケーション・サーバが用いられており，それらには機能面から「J2EE」[1] と呼ばれる Java API が多く採用されている．

また近年普及が進んでいる携帯情報端末向けプラットフォームである「Android」でも，Java による Android アプリが Google が運営する「Android マーケット」と呼ばれるソフトウェア販売サイトで有償，あるいは無償で多く提供されており，企業から一般ユーザまで開発人口が増加している．加えて，スマートフォンやタブレット PC をはじめとする多くの携帯情報端末には最新の HTML・CSS が実装された WEB ブラウザが搭載されており，それを利用した WEB アプリケーションの開発が Android の普及とともに高まると予想される．よって，WEB アプリケーションの開発を支援，あるいは開発された WEB アプリケーションのテスト工数を削減する支援ツールの需要も同時に高まると考えられる．

本研究ではそうした支援ツールとして，J2EE を用いた WEB アプリケーションのテスト工数を計測するテスト工数見積りツールの開発を行う．WEB アプリケーションのテスト工数見積りには，積算法と専門家による判定の二つの手法を用いる．本開発ツールにより，開発された WEB アプリケーションのテスト工数の見積りを円滑に行うことができ，時間的な側面から既存の WEB アプリケーションと比較することも可能になる．また付属的に，見積り対象となる WEB アプリケーションの実行時の WEB ブラウザ画面の入力系統の種類と個数を取得するので，GUI 単体のテストに利用することも可能である．

最後に，本報告書の以降の構成を説明する．第 2 章では本開発ツールで使用されるテスト工数見積り手法と，その他の既存の工数見積り手法について説明する．第 3 章では本開発ツールの見積り対象の WEB アプリケーションに用いられる J2EE と，その関連事項について説明する．第 4 章では本開発ツールによるテスト工数見積りの内部動作と，GUI 部の操作仕様について説明する．第 5 章では本開発ツールの WEB アプリケーションに対する適用実験と，その結果の考察について述べる．第 6

章では本開発ツールの評価に対する考察を行い，第 7 章で本研究のまとめと今後の課題を述べる．

2. ソフトウェアの工数見積り

IT 技術の進歩によってソフトウェア開発の規模が拡大するとともに，ソフトウェア開発の工数見積りの重要性もまた増大し，そのため過去様々な見積り手法が考案されている [2]．本章では，ソフトウェア開発の場でよく用いられる見積り手法について，本開発ツールにおける適用方法も踏まえて説明する．

2.1 見積り手法

見積り手法は以下のように大別することができる．

- 計算式によるコストモデル
  開発プロジェクトのコスト要因を変数としてコストモデルを構築し，見積りを行う手法である．初期は LOC（Lines Of Code）が用いられていたが，近年では Function Point 法が多く用いられる．この手法による見積りの有名なものには，COCOMO モデルがある．

- 専門家による判定
  複数の工数見積りの専門家の意見によって工数を見積る手法である．その際，Delphi 法などを用いて専門家の意見をまとめる必要があり，その工程によって見積り精度が変動する．

- 類似プロジェクトからの見積り
  過去の類似するプロジェクトから開発工数を見積る手法である．類似するプロジェクトが見つかれば見積りコストは減少するが，類似プロジェクトと開発プロジェクトとの条件の差異を見積りにどう影響させるかによって見積り精度が変動する．

- パーキンソンの法則の適用
  ソフトウェア開発においては，開発プロジェクトに使用可能なリソース（人月 [3] など）が見積られる工数に等しいという考え方．

- 契約価格による見積り
  顧客との契約交渉などによって決まる，顧客がそのプロジェクトに支払える価格が見積られる工数に等しいという考え方．

- 積算法
  開発プロジェクトで必要となる作業を洗い出し，作業毎の工数を積み上げていくボトムアップな手法である．作業の洗い出しにはWBS（Work Breakdown Structure）と呼ばれる構成図が多く用いられる．各作業に対する適切な工数割り当てが重要である．

見積り技術として考えると,「パーキンソンの法則の適用」と「契約価格による見積り」は受け入れ難い部分があるためあまり使用されない．ソフトウェア開発では,基本的には「計算式によるコストモデル」にその他の見積り手法を組み合わせて使用されることが多い．見積りは開発プロジェクトの計画段階から始められ，開発が進むにつれて徐々にその精度は向上する [4].

2.2 本研究での見積り対象

本研究では，テスト工数での工数見積りを対象とする．また，テスト対象のアプリケーションをJ2EEを用いたWEBアプリケーションとする．この条件下での工数見積り法として,「積算法」と「専門家による判定」の二つを利用した見積り手法を提案する．

具体的には，対象となるWEBアプリケーションから取得した入力系統の情報に積算法によって種類毎にテスト工数の割り当てを行うことで，WEBアプリケーションのテスト工数見積りを行う．入力系統の種類は特定のHTMLタグとして予め決まっているので，WBSを用いた洗い出しを行う必要は無い．テスト工数割り当てに用いられる入力系統毎のテスト工数の値は，専門家による判定によって決定した値を用いる．本開発ツールはWEBアプリケーションの開発者に使用されることを想定しているので，この時の専門家には本開発ツールの使用ユーザも含めてよいものとする．

3. J2EE (Java2 Enterprise Edition)

WEB アプリケーションでは，ユーザの要求に応じた動的なページ生成を行うために，クライアントからの要求は WEB サーバ内にある WEB コンテナの中で処理される．Java においては，その処理は WEB コンテナ内の JSP と Servlet によって行われる．本章では，見積り対象となる J2EE アプリケーションとそれに含まれる JSP について説明する．

3.1 J2EE とは

J2EE（Java2 Enterprise Edition）とは，Java における API の一つで，WEB 上で Java を動かすための API とその規約のセットである．2006 年より名称から「2」が外され，Java EE（Java Enterprise Edition）と新しく改名されている（バージョンは 5 であり，Java EE 5 と呼ばれる）．これは，Java の最も基本的な API である J2SE（Java 2 Platform Standard Edition）が Java のアップグレードによる大幅な仕様変更に伴い Java SE（Java Platform Standard Edition）に改名されたことによる．その後，Java EE は「Java EE 6」までバージョンが更新され，機能拡張や軽量化，開発容易性の向上などの改善が為されている．

API (Application Program Interface)

API（Application Program Interface）とは，あるプラットフォーム向けのソフトウェアを開発する際に使用できる命令・関数，またそれらを利用するためのプログラム上の手続きを定めた規約の集合のことである．API を利用することで，プログラマはその内部構造の詳細を知らずともその機能を利用したソフトウェアを容易に作成することができる．

API の基本的な考えは，多くのプログラムで頻繁に使用されるような共通の機能を API としてまとめて提供することである．こうすることで，各プログラマが別々に同じコーディングをすることの無駄が省略でき，コーディングがより簡潔になる．高レベルな API ほど簡潔なコーディングが可能になるが，代わりに低レベルな API が持つ柔軟性を失う．

3.2 JSP (Java Server Pages)

JSP（Java Server Pages）とは，HTML 内に Java のコードを埋め込んで実行させるための仕組みである．WEB コンテナは JSP に埋め込まれた Java のコードを処理し，実行結果を HTML として出力させる．それにより，クライアントの要求に応じた動的なページの生成が可能となる．JSP ファイルの拡張子は「.jsp」である．本開発ツールでは，主にこの JSP ファイルがテスト工数見積りの調査対象になる．

3.2.1 JSP と Java Script との違い

Java の実行結果を HTML として出力させる仕組みとして,「Java Script」がある．Java Script はコードを HTML 内に記述しておき，クライアントである WEB ブラウザ上で実行を行う．よって，Java Script のコードは HTML のソースコードから確認することができる．

一方，JSP は「Server」という文字が入っている通り，その実行は WEB サーバ側で行われる．その実行結果が HTML として WEB ブラウザに渡されるので，HTML のソースコードを見ても JSP の記述を見ることはできない．プログラムを WEB ブラウザが解釈し実行する Java Script と違い，JSP ではプログラムをサーバが解釈し実行するので，Java を導入していない端末からでもその WEB ページを利用することが可能になる．

3.2.2 JSP と Java Servlet との違い

JSP と似た機能を持つものとして「Java Servlet」がある．Java Servlet は Java クラスのコード内に HTML を記述する形式を取るのに対して，JSP のコードは HTML 内に Java のコードを断片的に含ませる形式を取るので，JSP のコードの外観は一見して HTML である．そのため，JSP ファイルに含まれる入力系統の HTML タグの種類と数が，テスト工数を見積もる際に必要となる要素になる．

JSP は，Java Servlet における Java プログラムのコーディングと WEB 画面のデザインとを同時に考えなければならない不便さを改善したものと言える [5]．これは JSP に，Java Servlet の WEB 画面のデザイン作業とコーディング作業とを切り離して考えることができるようにするという意図があるからである．また Java Servlet

と違い，JSP ではコンパイルもサーバ側の実行エンジンが行うのでコンパイル作業の必要がなく，その結果を保存し次の実行時に利用する仕組みになっているので効率的である．

4. WEB アプリケーションテスト工数見積りツール

本章では，本開発ツールが WEB アプリケーションのテスト工数見積りを行う際の内部動作と，GUI 部の操作仕様について説明する．内部のソースコードは，全て Python で書かれている．それを Python のオープンソースモジュールである「py2exe」で Windows 実行形式（exe ファイル）に変換したものが，本開発ツールである．

4.1 WEB アプリケーションのテスト工数見積り手法の提案

4.1.1 画面情報

WEB アプリケーションの実行時間を見積るためには，その WEB アプリケーション実行時の WEB ブラウザの画面情報が必要である．そのため，まず最初にその WEB アプリケーションの画面情報を含むファイル（以下，画面情報ファイル）の特定を行う．しかし，画面情報ファイルの場所や名前は明確に定まっておらず，対象の WEB アプリケーションによって様々である．よって，以下の条件で WEB アプリケーション内のディレクトリから目的のファイルを推定する．

条件 1: ファイル名に（screen，view，page）という単語を含むファイル

条件 2: ソースコードに 10 個以上のファイル名の記述を含むファイル

条件 3: 条件 1，2 を満たしたファイルの中で，最もファイル名を含むファイル

探索対象となるファイルの拡張子は（jsp，html，htm，xml，xhtml，java）の内のどれかで，それ以外の拡張子のファイルは探索対象とならない．
これら三つの条件は，

> 画面情報ファイルには，それぞれの画面に配置されるファイル名が記述されているため，ソースコードに含まれるファイル名は多くなる．

という推測に基づいたものである．これら三つの条件に当てはまるファイルが一つも該当しなかった場合（つまりは画面情報が見つからなかった場合）は，後述する別の方法でテスト工数の見積りを行う．その場合に見積られるテスト工数の値は，画面情報ファイルが見つかった場合の値よりも，特別な場合を除き正確ではなく大雑把な値になる．

4.1.2　テスト工数見積り

テスト工数見積りは，画面情報ファイルと推定されたファイルのソースコードに記述されている全てのファイルを対象に行う．その際，ファイル名が重複していても構わないものとする（別々の画面で同じファイルが使用されているため）．それらのファイルのソースコードから入力系統の HTML タグの種類とその属性，個数を取得し，事前に設定しておいたテスト工数の値を入力系統毎に掛け合わせて合計することで，最終的なテスト工数を見積る [6].

入力系統

テスト工数を見積る入力系統としては，HTML タグの入力フォームに定められる，

- <input>タグ
- <select>タグ
- <textarea>タグ
- <button>タグ

の四つを対象とする．以下で，それぞれのタグについて詳しく説明する．

- <input>タグ
  type 属性によって様々なフォーム部品を画面に作成する要素である．テキストボックスを作成する部品の type 属性には，フォームの大きさを設定する size 属性と，最大文字数を設定する maxlength 属性があり，この二つの属性から最大テキスト量を計算する．具体的には，maxlength 属性を持つ場合はその値を最大テキスト量として使用し，持たない場合は size 属性から最大テキスト量を算出して使用する．HTML が HTML5 に更新されて，電話番号や日時などの入力対象に応じたテキストボックスを作成する type 属性がいくつか追加されたが，ここではテスト工数に関係がある type 属性についてだけ説明する．
  - text：一行のテキストボックスを作成する属性．size 属性と maxlength 属性を持つ
  - password：パスワード入力欄を作成する属性．size 属性と maxlength 属性を持つ

  - checkbox：チェックボックスを作成する属性
  - radio：ラジオボタンを作成する属性
  - image：画像ボタンを作成する属性

- <select>タグ

  セレクトボックスを画面に作成する要素．option 属性によって選択肢が作成される．<select>タグのテスト工数は，<select>タグ自身の数と option 属性の数によってそれぞれ計算される．

  - option：セレクトボックスに選択肢を配置する属性

- <textarea>タグ

  複数行のテキストボックスを画面に作成する要素である．cols 属性でテキストボックスの最大文字数を，rows 属性でテキストボックスの行数を指定できる．<textarea>タグのテスト工数は，これら二つの値を掛けた最大テキスト量を出すことで計算される．<textarea>タグは，type 属性が text の<input>タグとしてまとめて計算され，文字数もこれに加算される．

  - cols：テキストボックスの最大文字数を指定する属性．指定しない場合，初期値は自動的に 20 となる
  - rows：テキストボックスの行数を指定する属性．指定しない場合，初期値は自動的に 2 となる

- <button>タグ

  ボタンの部品を画面に作成する要素である．type 属性によってボタンの種類を指定できるが，種類によるテスト工数への影響はない．

これらの入力系統をファイルのソースコードから発見すると，プログラムはその入力系統の種類と属性を GUI に渡す．

画面情報ファイルが推定できなかった場合

以下のような場合，画面情報ファイルは 4.1.1 節に示す条件では推定できない．

- 画面情報ファイルのソースコードにファイル名の記述が少ない場合

- 画面情報ファイルの名前に（screen，view，page）が含まれていない場合

そういった場合，プログラムは対象となるWEBアプリケーションのディレクトリ内にあるファイルのソースコードを全て調査し，その情報だけでテスト工数見積りを行う．この方法では画面情報が無いため，複数の画面における同じファイルの重複使用を検出することはできない．そのため，この場合に見積もられるテスト工数の値は，対象のWEBアプリケーションで重複使用されているファイルが存在しない場合を除き，画面情報ファイルが推定できた場合のテスト工数の値よりも正確な値ではなくなる．

ただこの値は，本ツールによる探索がWEBアプリケーション実行時に読み込まれるファイルを正確に全て網羅できているかを確認する作業に有用である．なぜなら，画面情報が有る状態で見積られたテスト工数から複数の画面間で重複しているファイルのテスト工数を差し引いた値は，画面情報が無しの状態で見積られたテスト工数と一致している筈であるからである．これらが一致していなかった場合，プログラムには何らかの画面情報の取得もれがあることを示している．よって，この値をWEBアプリケーションのテスト工数見積り結果の評価に用いる．

4.2 GUI

本開発ツールを起動すると，図4.1に示すGUI画面1が立ち上がる．GUI画面1の各入力の機能は以下の通りである．

- 入力フォーム：対象となるWEBアプリケーションの指定（絶対パス）
- Browseボタン：ブラウズして対象となるWEBアプリケーションを直接指定
- Estimateボタン：テスト工数見積りの開始
- Optionボタン：それぞれの入力系統に対するテスト工数の設定

4.2.1 WEBアプリケーションの指定

テスト工数を見積りたいWEBアプリケーションのディレクトリの絶対パスを，入力フォームに入力することで見積り対象を指定する．またBrowseボタンをクリックすることで，ディレクトリ構造をブラウズし，ディレクトリを直接選択することも可能である．


TestKun
Target software directory:
Browse...
Estimate
Options...
テスト工数見積君 © 2011-2012 / SEL@KIT
Version 0.1

図 4.1　GUI 画面 1

4.2.2 テスト工数の見積り

Estimate ボタンをクリックすることで，テスト工数の見積りが行われる．プログラムが対象となる WEB アプリケーションから入力系統の情報を取得し，その情報に従い GUI がそれぞれの入力系統に対して設定されたテスト工数を掛け合わせ，それらを合計したテスト工数の値が図 4.2 の GUI 画面 2 のように見積り結果として表示される．結果の上半分は見積られた WEB アプリケーションの総テスト工数を，下半分は各画面毎のテスト工数を個別に表示するものであるが，その機能は未実装であり現状では共に WEB アプリケーションの総テスト工数が表示される．現時点では画面情報ファイルから各画面の情報を区別して取得することが困難であり，この機能の実装は今後の課題である．

GUI 画面 2 の各入力の機能は以下の通りである．

- Save ボタン：見積り結果を保存する
- Quit ボタン：プログラムの終了
- Back ボタン：GUI 画面 2 を閉じる

4.2.3 入力系統のテスト工数設定

Option ボタンをクリックすることで，図 4.3 の GUI 画面 3 が表示される．この画面では，それぞれの入力系統毎のテスト工数が設定できる．本開発ツールは分単位の見積りを想定しており，各入力系統の入力フォームには 0 から 100 までの数字が入力できる．

GUI 画面 3 の各入力の機能は以下の通りである．

- OK ボタン：入力したテスト工数設定を決定する
- Cancel ボタン：テスト工数設定を決定せずに GUI 画面 3 を閉じる
- Effort per item：それぞれの入力系統毎にテスト工数を設定できる
- Effort per character：Text box と Password box のみ．文字一つ毎のテスト工数を設定できる
- Effort per option：Select のみ．option 属性一つ毎のテスト工数を設定できる

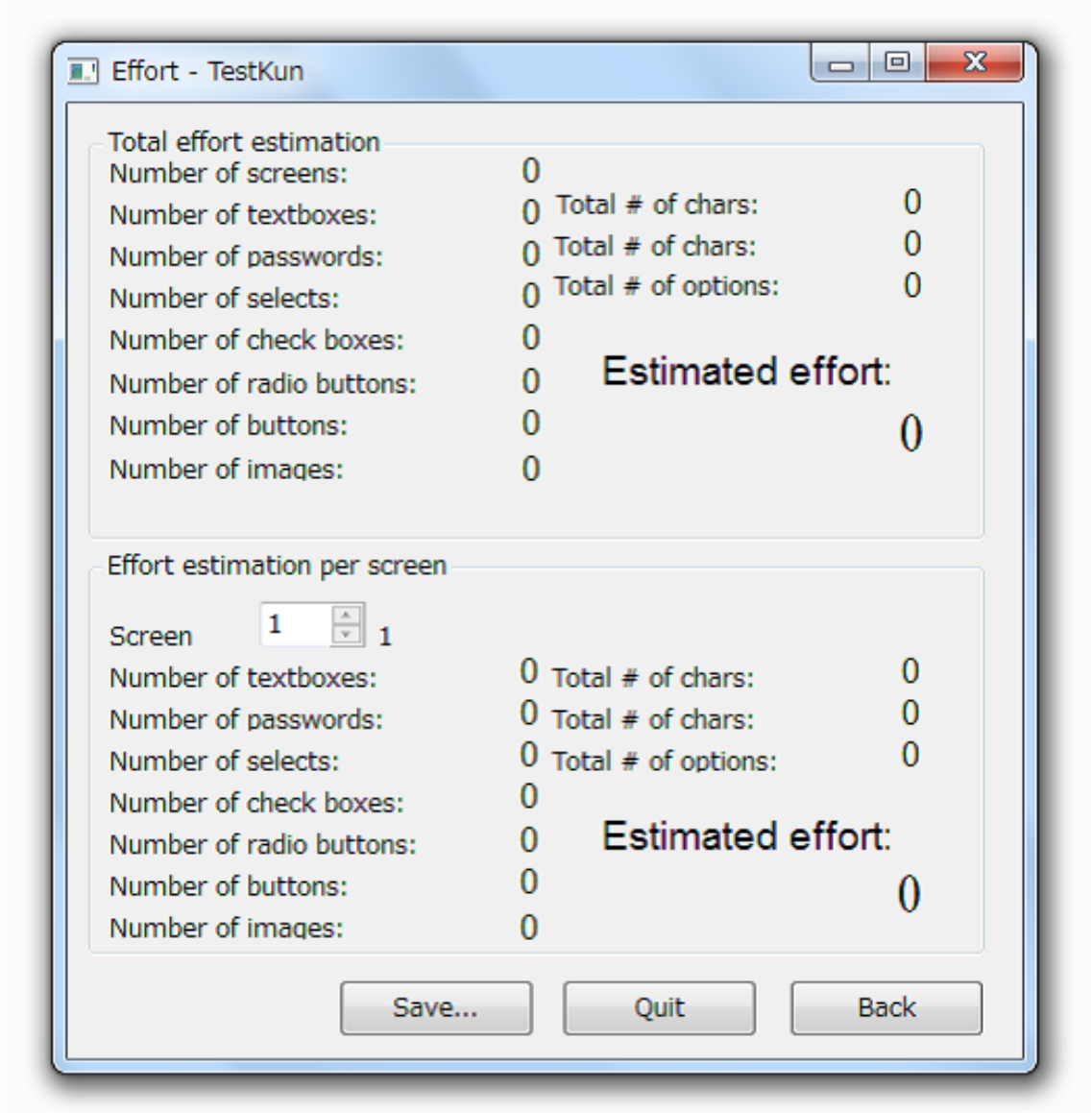
Effort - TestKun
Total effort estimation
Number of screens: 0
Number of textboxes: 0
Number of passwords: 0
Number of selects: 0
Number of check boxes: 0
Number of radio buttons: 0
Number of buttons: 0
Number of images: 0
Total # of chars: 0
Total # of chars: 0
Total # of options: 0
Estimated effort:
0
Effort estimation per screen
Screen 1 1
Number of textboxes: 0
Number of passwords: 0
Number of selects: 0
Number of check boxes: 0
Number of radio buttons: 0
Number of buttons: 0
Number of images: 0
Total # of chars: 0
Total # of chars: 0
Total # of options: 0
Estimated effort:
0
Save...
Quit
Back

図 4.2　GUI 画面 2

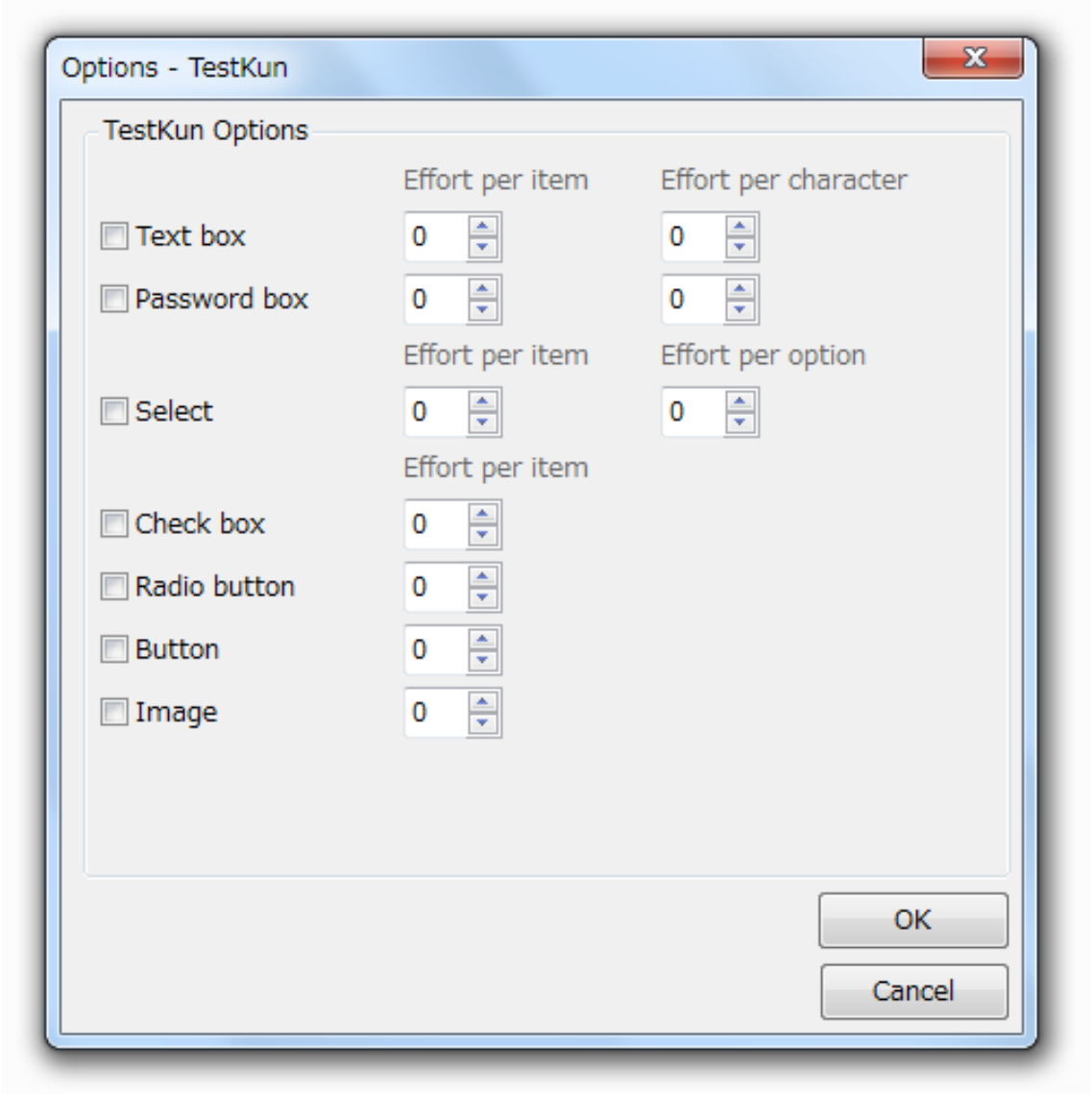
Options - TestKun
TestKun Options
Effort per item
Effort per character
Text box
0
0
Password box
0
0
Effort per item
Effort per option
Select
0
0
Effort per item
Check box
0
Radio button
0
Button
0
Image
0
OK
Cancel

図 4.3　GUI 画面 3

4.2.4 改善点

GUI 部には，4.2.2 節で示したように各画面のテスト工数見積り結果が表示できないという点の他に，ファイル数の多い WEB アプリケーションを見積り対象とした時に，Estimate ボタンをクリックした後（テスト工数見積りの最中），動作が一時的にフリーズする（したかのように見える）という問題点がある．これはファイル探索に時間がかかることが原因であるが，そのことをユーザに伝える機能を実装していないため，プログラムがフリーズしたのではないかとユーザに不安を与えかねない．プログラムの状態がビジーであることと，今どれだけテスト工数の見積りが終了したかをユーザにアナウンスする機能をユーザインタフェースの面から実装する必要がある．

5. WEB アプリケーションへの適用実験

本章では，本開発ツールの WEB アプリケーションへの適用実験を行う．実験対象は，J2EE を用いた WEB アプリケーションの設計サンプルとして Oracle から提供されている「The Java Petstore 」[7]（以下，単に Java Petstore）の Ver.1.3.2 を対象とする．実験結果の評価に用いる Java Petstore に含まれる入力系統の情報の真値を表 5.1 に示す.

5.1 見積り値の評価方法

実験によって見積られたテスト工数値は，4.1.2 節で示した方法によって評価される．必要となる値は以下の二つである．

(1) ファイル毎の入力系統の合計

(2) 重複ファイルの入力系統の合計

まず（1）を，画面情報ファイルが推定できなかった場合と同じく，Java Petstore に含まれる全てのファイルから入力系統の情報を検索することによって取得する．

次に，画面情報から複数の画面で重複使用されているファイルを調べる（Java Petstore は J2EE の設計サンプルであるので，画面情報が非常にわかりやすく重複ファイルの取得は比較的容易である）．そしてそれらのファイルから（2）を求める．この際，複数回使用されているファイルは，使用されている回数分の入力系統を足し合わせる必要がある．

こうして求めた（1）と（2）を更に入力系統毎に足し合わせた値が，本開発ツールによって見積られた入力系統の数と一致しているかどうかを調べる．

5.2 実験結果

本開発ツールを Java Petstore へ適用した結果と，その値を（1）,（2）の合計と比較した結果を表 5.2 に示す．また Java Petstore のテスト工数の見積り結果を図 5.1 に，その時のテスト工数の設定を表 5.3 に示す.

表 5.1　Java Petstore の入力系統

入力系統	合計	考慮される属性	合計
text	72	合計文字数	2657
password	2	合計文字数	35
select	19	合計 option 数	73
check box	5	—	—
radio button	0	—	—
button	0	—	—
image	44	—	—

表 5.2　実験結果：入力系統の比較

入力系統	ファイル毎の入力系統合計	重複ファイルの入力系統合計	合計	Java Petstore の入力系統見積り
text	20	52	72	72
合計文字数	913	1744	2657	2657
password	2	0	2	2
合計文字数	35	0	35	35
select	11	8	19	19
合計 option 数	49	24	73	73
check box	5	0	5	5
radio button	0	0	0	0
button	0	0	0	0
image	19	25	44	44

表 5.3　実験時のテスト工数設定

入力系統	工数値	考慮される属性	工数値
text	5	Effort per character	1
password	3	Effort per character	1
select	3	Effort per option	1
check box	1	—	—
radio button	1	—	—
button	1	—	—
image	1	—	—

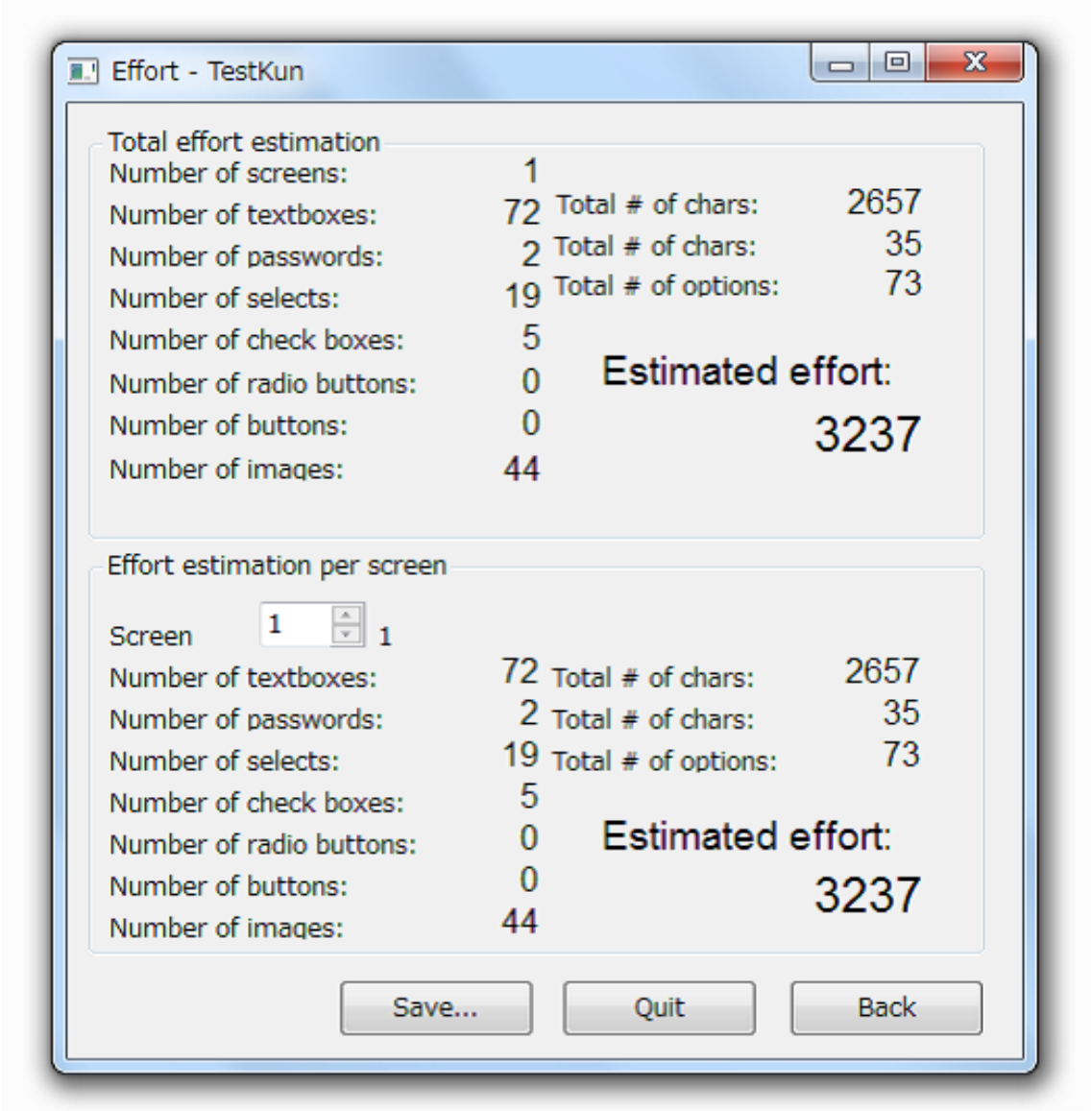

図 5.1　Java Petstore のテスト工数の見積り結果

表 5.2 の右二列の値を見ると，確かに両列のそれぞれの入力系統の値が一致していることがわかる．よって，本開発ツールによる画面情報の取得漏れがないことと，画面情報ファイルが正しく特定できていることが確認できた．見積られた Java Petstore のテスト工数は図 5.1 の「Estimated effort」に示された値であり，テスト工数設定を変更することでこの値が変動する．

表 5.3 に示す工数値は，バルテス株式会社 第一ソフトウェアテスト部 部長 角田誠氏と，同じく技術開発部 部長 石原一宏氏より頂いた助言を元に設定した．

6. 考察

本ツールの有効性

元々，本開発ツールはWEBアプリケーションのテスト作業の工数を見積るための支援ツールとして開発された．しかし，個々のWEBアプリケーションによってその画面情報の記述方法は一定しておらず，各画面毎の情報と画面遷移の情報を安定して取得することが困難だったことから，見積り方法を見直し，見積り対象をWEBアプリケーションの全ての画面（あるいは全てのファイル）とした．それによりWEBアプリケーション全体の入力系統のテスト工数合計の見積りが可能になった．しかし，画面遷移の情報が必要となる機能テストやシナリオテストなどの，WEBアプリケーションの使用ユーザの要求を考慮したテスト作業は補えなくなった．

よって，本開発ツールが有効なのは，開発するWEBアプリケーションの入力系統の実装，単体テスト，及び他のWEBアプリケーションに対する比較値の提供といった場面に限定されるということがわかった．

妥当性の検証

最後に，本研究の妥当性について以下に示す．

- プログラムのバグ
  本開発ツールのソースコードにバグがある可能性がある．

- HTMLタグの記述への対応
  HTMLタグの記述の仕方によっては入力系統として認識されずに取得漏れが発生し，正確なテスト工数が見積れなくなってしまう可能性がある．

- 画面情報ファイル特定のための条件
  4.1.1節で示した条件が，画面情報ファイルを特定するためのものとして十分に効果を発揮しない可能性がある．

- 個々のWEBアプリケーションのファイル構造への対応
  対象のWEBアプリケーションのファイル構造（ファイル名，ファイル内容な

ど）によっては 4.1.1 節に示す条件が十分な効果を示さず，画面情報ファイルの推定が上手く行かず正しくテスト工数を見積れない可能性がある．

7. 結言

本研究では，J2EE を用いた WEB アプリケーションの実行時間を，画面情報に従いファイルのソースコードから入力系統の種類，属性，個数を取得することによって見積りを行うテスト工数見積りツールの開発を行った．それによって WEB アプリケーションの WEB ブラウザ画面全体のテスト工数の見積りと入力系統の情報の取得が可能になった．しかし，本開発ツールはリンクによる画面遷移やユーザの入力時における状況等を考慮していないため，機能テストやシナリオテストといったユーザ要求に応じたテスト工数の見積りは行えないという問題が判明した．

そのため，本ツールは入力系統の個数と WEB アプリケーション全体のテスト工数を提供することで，同等のサイズの WEB アプリケーションの開発，及び入力系統のテストには有用であるが，このツールだけでテスト工数の見積り作業の全てを補うことは困難であるということがわかった．この問題の解決には，画面情報ファイルから各画面の情報を区別して取得することと正確な画面遷移の情報が必要となる．

今後の課題として，本開発ツールの改良されるべき点を内部動作と GUI 部に分けて以下に示す．

- (内部) WEB アプリケーションの各画面情報の識別
- (内部) WEB アプリケーションの画面遷移の情報の取得
- (GUI) 各画面毎のテスト工数の見積りの表示
- (GUI) テスト工数の見積り時におけるプログラムの状態のユーザへの提示

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本報告書の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学部門水野修准教授，加えて，本報告書執筆にあたり貴重な助言を多数頂きました，バルテス株式会社 第一ソフトウェアテスト部 部長 角田誠氏，同じく技術開発部 部長 石原一宏氏に厚く御礼申し上げます．

また本学情報工学専攻 平田幸直先輩，出原真人先輩，中井道先輩，川本公章先輩，梁軍偉先輩，情報工学課程 向井弘記君，小野木祐太君，松村好剛君，西村祐輔君，

学生生活を通じて筆者の支えとなった家族や友人に心から深く感謝致します.

参考文献

[1] Oracle, Java EE at a Glance, Java EE at a Glance（オンライン），入手先 ⟨http://www.oracle.com/technetwork/java/javaee/overview/index.html⟩（参照 2012-02-14）.

[2] N.E. Fenton, Software Metrics: A Rigorous and Practical Approach, 2nd edition, International Thomson Computer Press, Boston, MA, USA, 1996.

[3] フレデリック・P・ブルックス，人月の神話　原著発行 20 周年記念増刷版 (滝沢徹，牧野祐子，富澤昇 (訳))，ピアソン・エデュケーション，2002.

[4] 大筆　豊，“ソフトウェアのコスト見積り技術，” 情報処理，vol.33，no.8，pp.906–911，1992-08-15.

[5] 三谷純，新人 SE のための Java 講座 Tomcat を使う「JSP プログラミング」，連載：Tomcat を使う「JSP プログラミング」 第 1 回（オンライン），入手先 ⟨http://www.atmarkit.co.jp/fjava/rensai/jsp01/jsp01.html⟩（参照 2012-02-14）.

[6] 今井 義治，太田 義勝，鈴木 秀智，“ソフトウェア開発工程における GUI 部分の工数見積について，” 情報処理，vol.69，no.1，pp.1.331–1.332，2007-03-06.

[7] Oracle, Java Pet Store, Java Pet Store（オンライン），入手先 ⟨http://java.sun.com/developer/releases/petstore/⟩（参照 2012-02-14）.