社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

画面遷移とデータベース処理を考慮したトランザクションファンクション識別手法の詳細化と実装

枝川　拓人†　　赤池　輝彦†　　肥後　芳樹†　　楠本　真二†

† 大阪大学大学院情報科学研究科〒 560-8531 大阪府豊中市待兼山町 1-3
E-mail: †{t-edagaw,t-akaike,higo,kusumoto}@ist.osaka-u.ac.jp

あらまし　ファンクションポイント法とは, ソフトウェアが持つ機能数を基に, 規模を計測する手法である. 計測されたファンクションポイントは主に工数見積りに使用されるが, その際, 過去のデータが十分そろっているか否かが見積りの精度を決定する. したがって, 開発の終了したソフトウェアからファンクションポイントを計測する手法の開発は, 過去のデータを効率よく蓄積する上で有効である. 我々はこれまでに, Web アプリケーションのソースコードからファンクションポイントを自動計測する手法を提案してきた. 本稿では, この既存手法において問題となっていた入出力機能の識別に関する改善手法を提案する. また, 既存手法に改善手法の一部を加えた手法を実装したツールを Web アプリケーションに対して適用し, 手法の有用性を確認する.

キーワード　見積り, 静的解析, ファンクションポイント, Web アプリケーション, SQL

Refinement of Transactional Functions Extraction Based on Screen Transitions and Database Accessses

Takuto EDAGAWA†, Teruhiko AKAIKE†, Yoshiki HIGO†, and Shinji KUSUMOTO†

† Graduate School of Information Science and Technology, Osaka University
1-3, Machikaneyama-cho, Toyonaka, Osaka, 560-8531, Japan
E-mail: †{t-edagaw,t-akaike,higo,kusumoto}@ist.osaka-u.ac.jp

Abstract　Function Point(FP) is one of the software size metrics and counts the amount of functionality which information system provides to a user. For using FP to estimate the development effort with high accuracy, rich data is required in a project metric repository. For this purpose, we had proposed a automatic FP measurement method from Web Application. In this paper, we improve the method, focusing on the identification of Transaction Function, and implement the tool for measuring FP with the method, and adopt the tool to a Web application.

Key words　Estimation, Static Analysis, Function-Point,Web Application, SQL

1. はじめに

近年, ソフトウェアはますます大規模化, 複雑化, 多様化してきており, その上で開発期間の短縮化が求められるようになってきている. したがって, 高品質なソフトウェアを効率良く開発するために, 開発計画の下で開発プロセスの全工程を系統づけて管理する必要性が高まってきている.

ソフトウェアプロジェクトの計画の際に重要な情報となる開発工数は, 通常, ソフトウェアの規模を基にして予測される. このソフトウェアの規模を予測する手法として, ファンクションポイント (FP) 法 [1] が用いられている.

FP 法は, ソフトウェアの機能的な規模を見積る手法として, 1987 年に Albrecht によって提案された. 現在, これをベースに IFPUG 法 [2] が考案され, 広く実用されている. FP 法は, 要求仕様書や設計仕様書等からソフトウェアの機能要件だけを抽出して定量的にソフトウェアの規模を計測する手法である.

FP を基にして精度の高い見積りを行うためには, そのソフトウェア開発組織で過去に開発されたソフトウェアの FP や開発工数の実績値が基礎データとして充実していることが条件となるため, 基礎データが充実していない場合, 過去のプロジェクトに対して FP を計測する必要がある. しかしこれを従来のように仕様書を基にして人の手によって計測する場合, (1) 余分なコストがかかる, (2) 仕様書が残存しない場合計測できない, (3) 開発途中で発生した仕様変更が仕様書に反映されていない場合正確な FP を計測できない, といった問題が発生する.

したがって, 開発の終了したソフトウェアから自動で FP を

計測する手法の開発は, 効率よく過去のデータを蓄積する上で非常に有用である. そこで, 我々はこれまでに, 特定の構成で開発された Web アプリケーションのソースコードから FP を自動計測する手法を提案してきた.

本稿では, 既存手法の問題点となっていた入出力機能の識別手法の改良案を提示する. また, 既存手法に改良手法の一部を加えた手法をツールとして実装し, Web アプリケーションに適用することで提案手法の評価を行う. 評価では, (1) ツールによる計測結果と熟練者による計測結果の比較, (2) 同一仕様から独立に開発された複数のシステムに対するツールの計測結果間での比較, を行い, 手法の有用性と実装への非依存性を確認する.

2. 背景技術

2.1 IFPUG 法

本研究では, 数多くの FP 法の中で主流技法となっている, IFPUG 法を基にして FP を計測する.

IFPUG 法では, 計測する機能をデータファンクションとトランザクションファンクションの 2 種類に分類する. そして, この 2 種類の機能をアプリケーションから抽出し, 抽出された各機能に対してその機能の複雑さを基にして FP を計算し, それらを合計することで, アプリケーション全体の FP を計測する.

このようにして計測された FP は未調整 FP と呼ばれ, さらにシステム特性を考慮することで最終 FP が得られるが, システム特性等をソースコードから得るのは困難であるため, 本研究では計測する FP を未調整 FP としている.

以下に, IFPUG 法による FP 計測の主要プロセスである, データファンクションとトランザクションファンクションの計測方法を紹介する.

データファンクションの計測

データファンクション (DF) とは, ユーザが認識できる論理的なデータのまとまりである.

DF は, 以下の 2 種類に分類される.

- 内部論理ファイル (ILF)
- 外部インターフェイスファイル (EIF)

ILF は, 計測対象のアプリケーション内で更新されるデータの集合であり, EIF は, 計測対象のアプリケーションによってデータが更新されることがなく, 参照のみされるデータの集合を意味する.

DF の複雑さは, 以下の 2 つのパラメータを用いて, 低・中・高の 3 段階に決定される.

- データ項目数 (DET)
- レコード種類数 (RET)

DET は, DF 中のユーザが認識できる論理的な項目数を, RET は, DF 中の異なる意味合いをもつ集合の個数を意味する.

トランザクションファンクションの計測

アプリケーションに対するデータの出入りを伴う処理をトランザクションファンクション (TF) という.

TF は, 以下の 3 種類に分類される.

- 外部入力 (EI)
- 外部出力 (EO)
- 外部照会 (EQ)

EI は, ユーザからのデータ入力によって DF の更新を行うことを主目的とする処理である. EO は, ユーザへのデータ出力を主目的とする処理のうち, (1) 処理ロジックに数式処理または演算を含むもの, (2) 出力データに派生データ (計算や条件判断など何らかの加工を必要とするデータ項目) を含むもの, (3) 処理が内部論理ファイルを更新するもの. のいずれかの条件を満たすものである. EQ は, ユーザへのデータ出力を含む処理のうち, EO ではないものを意味する.

TF の複雑さは, 以下の 2 つのパラメータによって, 低・中・高の 3 段階に決定される.

- データ項目数 (DET)
- 関連ファイル数 (FTR)

DET は, TF が入出力するデータの項目数を, FTR は, TF が参照, 更新する DF の数を意味する.

2.2 MVC モデル

本研究では, MVC モデルで開発された Web アプリケーションを適用対象としている.

MVC モデルとは, Model, View, Controller の 3 種のモジュールでシステムを構成する Web アプリケーションの開発モデルである. Controller がユーザからリクエストを受取り, Model が機能を実行し, View がユーザに画面を提示する, という構成になっている.

MVC モデルを提供するフレームワークとしては, Struts [3] が広く実用されている.

3. 既存手法 [4]

3.1 方　　針

FP をソースコードから計測する際, もっとも重要となるのは, 機能がソースコード上にどのような形で実装されているかを決定することである. 既存手法では, 適用対象を“ MVC モデルで開発された SQL を使用する Web アプリケーション ”に限定した上で, DF や TF を以下のように定義している.

DF : データベース上のテーブル
TF : 画面遷移時に実行される SQL の集合

DF はシステムによって使用されるデータの論理的集合である為, データベースのテーブルを対応させている. また, TF はユーザがシステムに対して行う入出力処理である為, Web アプリケーションにおいては画面遷移時に実行されるデータベースアクセス (SQL) の集合を対応させている.

3.2 計測手法

既存手法は, 以下の手順で FP を計測する (図 1 参照).

Step1 : 各画面遷移で実行される SQL を抽出する
Step2 : SQL から DF を識別する
Step3 : SQL から TF を識別する

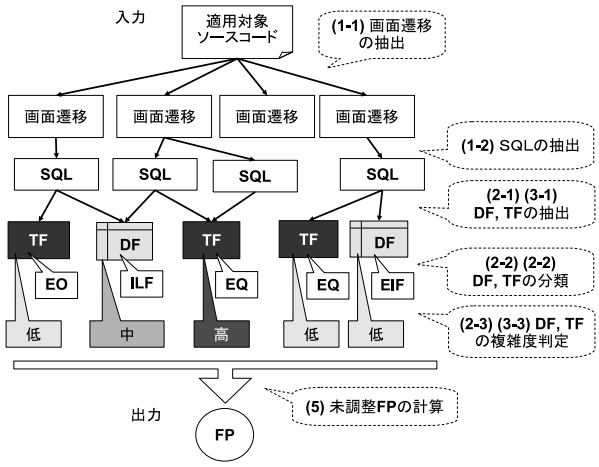

図 1　計 測 手 順

Step4：未調整 FP を計算する

Step1：SQL の抽出

まず, MVC モデルを構成しているフレームワークの設定ファイル（Struts では struts-config.xml）を解析して, 画面遷移時に初めに呼び出されるメソッドを取得する. そして, このメソッドを起点としてソースコードを解析し, 各画面遷移で実行される SQL を収集する.

Step2：DF の識別

まず, Step1 で収集した SQL から DF を抽出する. DF はテーブルに相当するため, SQL で参照されているテーブルを抽出すればよい. しかし, システムの実装上の理由のみで作られているようなテーブル（例：割り振る ID を管理するためだけのテーブル）は, ユーザに認識されないため DF とはみなされないので, このようなテーブルは排除する必要がある. このテーブルは現段階では自動で除去することは出来ないため, ユーザが手動で除去する必要がある.

次に, 以下の手順で DF を ILF, EIF に分類する.

RuleD1: データベースを更新する SQL(insert や update, delete など) でアクセスされている場合, ILF と判定する

RuleD2: RuleD1 を満たさない場合, EIF と判定する.

最後に, DF の複雑度を判定する. DF の複雑度決定要素である DET, RET は以下のように計算する

- DET = テーブルの属性の数,
- RET = 1.

RET はデータベースの正規化レベルが高いほど 1 に近づくと考えられる. 我々は適用対象の正規化レベルが高いことを仮定して, RET を 1 として扱うこととした.

Step3：TF の識別

まず, Step1 で抽出した各画面遷移ごとの SQL 集合を TF として抽出する.

次に, 以下の手順で TF を EI, EO, EQ に分類する.

RuleT1: データベースを更新する SQL(insert や update, delete など) を一つでも持つ場合, EI と判定する,

RuleT2: RuleT1 を満たさず, データベースから取得したデータに算術演算がなされていた場合, EO と判定する,

RuleT3: RuleT1, RuleT2 を満たさない場合, EQ と判定する

最後に, TF の複雑度を決定する. TF の複雑度決定要素である DET, FTR は以下のように計算する.

- DET = 入力項目の数＋ SQL で取得したデータ項目の数
- FTR = SQL で参照もしくは更新したテーブルの数

Step4：未調整 FP の計算

機能種別と複雑度から各機能の FP を計算し, それらを合計することでシステム全体の未調整 FP を得る.

4. 改 良 手 法

本研究では, 既存手法における TF 識別に関する改善を行った. 既存手法における TF 識別に関する問題点は, 以下の二つに分割できる.

（1） TF の抽出手法
（2） 抽出した TF の分類手法

4.1 トランザクションファンクション抽出手法

既存手法においては, TF は“ 画面遷移時に実行されうる全 SQL の集合 ”と定義されている. しかし, この定義では一つの画面遷移からは高々一つの TF しか抽出できない. つまり, ユーザの入力値によって実行する機能が変化する画面遷移のように, 一つの画面遷移に複数の機能が実装されている場合でも, まとまった一つの機能として抽出されてしまう.

そこで, TF の定義を“ 画面遷移時に同時に実行されうる全 SQL の集合 ”と修正した. この定義であれば, 条件によって実行する機能が違う画面遷移からでも, 複数の機能を抽出することができる.

このように定義を変更した場合, いかにして同時に実行されうる SQL の集合を得るかが問題となる. そこで, 既存手法で単純に実行される SQL を抽出していた Step(3.2 章 Step1 を参照）に改良を加え, SQL Tree という条件分岐構造を保持するデータ構造に, 画面遷移中に実行されうる SQL を保持するようにした. そして構築された SQL Tree を解析することで, 「同時に実行される SQL の集合」を TF として抽出する.

(1) SQL Tree の構築

SQL Tree は三種類のノードを持つ.

- Code Node: ソースコードに対応するノード,
- Branch Node: 条件分岐に対応するノード,
- SQL Call Node: SQL 呼び出しに対応するノード.

Code Node は, Branch Node, SQL Call Node を子として持つことが出来る. Branch Node は, Code Node を子として持ち, SQL Call Node は子を持たない.

図 2 に SQL Tree の構築例を示す. この例では, まずコード全体に対応する Code Node(a) がルートとして構築される. そして, (a) が対応するコードの直下には一つの if-else 文が存在するため, (a) の下に Branch Node が作られ, 条件分岐先の Code

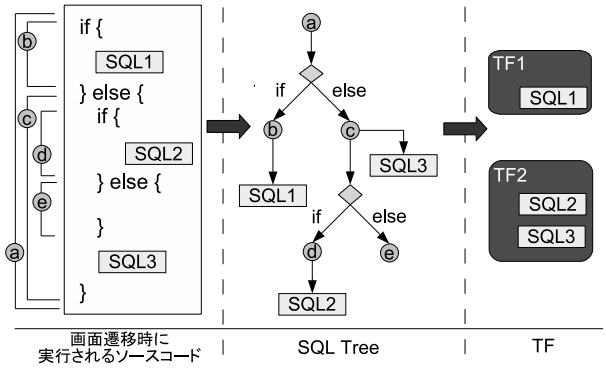

図 2　SQL Tree の構築と TF 抽出の例

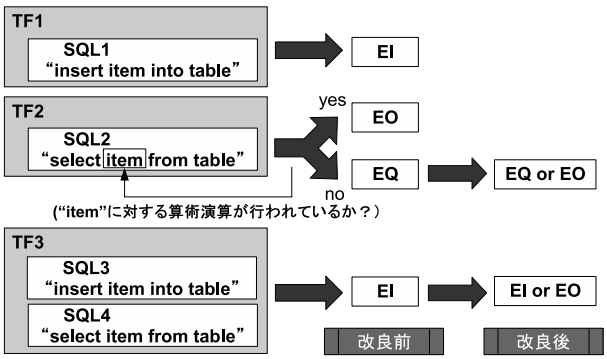

図 3　TF 分類の例

Node(b), (c) が, Branch Node の下に作られる．(b), (c) に対応するコードでは, それぞれ SQL1, SQL3 が呼ばれているので, (b), (c) の下に SQL Call Node が生成される．(c) に対応するコードの中にある if-else 文に関しても同様に処理する．

(2) SQL Tree からの TF 抽出

このようにして構築した SQL Tree から TF を抽出する．改良手法では TF は「画面遷移時に同時に実行されうる全 SQL の集合」と定義したので, SQL Tree の中で同時に実行されうる SQL の集合を見つければよい．図 2 の SQL Tree には 3 つの SQL 呼び出しがある．この内 SQL2 と SQL3 は同時に実行される可能性があるが, SQL1 はこれらと同時に実行されることはない．そのため, この SQL Tree からは SQL1 を持つ TF1 と, SQL2 と SQL3 を持つ TF2 が抽出される．

4.2　トランザクションファンクション分類手法

既存手法における TF 分類手法 (3.2 章 Step3 参照) には, 以下の二つの問題点がある．

- EO もデータベースを更新する場合がある．すなわち, RuleT1 では EI と確定することはできない．
- 算術演算がなされない出力でも, EO の可能性はある．すなわち, RuleT3 では EQ とは確定できない．

つまり, RuleT1 には“ EI or EO ”の, RuleT3 には“ EQ or EO ”の判定を追加する必要がある．改良前と改良後の, TF 分類の例を図 3 に示す．

EI or EO

出力も入力も行うような機能の場合, EI か EO かの判定をする必要がある．IFPUG 法では, このような機能の判定手法として「入力と出力のどちらが主目的か」という基準が用意されている．すなわち, 入力が主目的の機能であれば EI, 出力が主目的の機能であれば EO と判定する, ということである．

この「どちらが主目的か」という判断を自動で行うのは非常に困難であるため, 以下のようなメトリクスを用いて, 間接的に判定する方法を検討している．

- SQL の数
- SQL を呼び出した文の条件分岐構造
- SQL で取得・入力するデータの数
- SQL で取得・入力するデータへの参照回数

まず, 「SQL の数」が多いほど目的度が高いと考えられる．例えば, select 文が 5 回, insert 文が 1 回呼ばれるような機能では, 出力が主目的である機能の可能性が高い．次に, 「SQL を呼び出した条件分岐構造」が, 目的の強さに関連すると考えられる．より SQL を呼び出す条件が厳しいほど, 主目的の可能性は低くなる．また, 「SQL で取得・入力するデータの数, そのデータへの参照回数」の数が多いほど目的度が高いと考えられる．

これらのメトリクスを用いて, 入出力のどちらが主目的かをどのように判断するかは検討中である．多くの実験を重ねた上で, メトリクスの用い方を判断する必要がある．

EQ or EO

出力のみを行うような機能では, EQ か EO かを判断する必要がある．IFPUG 法ではこのような場合,

- 数式処理もしくは演算を行う
- 派生データを生成する

のどちらかの条件を満たせば EO と判定し, どちらも満たさない場合 EQ と判定するように規定されている．既存手法では, 前者は判定しているが, 後者は判定していない．つまり, 派生データ生成の有無を判定する必要がある．

派生データの生成を判定する手法としては, 「データベースから取得したデータと, ユーザに提供するデータを比較する」という手法が考えられる．すなわち, データベースから取得したデータ以外のデータをユーザに提供している場合, 何らかの派生データを生成していると判断するというものである．

この手法に関しても, 実験を積み重ねていく中で詳細化を進める必要がある．

5.　実　　装

既存手法に, TF 抽出手法の改良を加えた手法をツールとして実装した．TF 分類手法の改良に関しては, 詳細化が十分ではないこともあり, 実装には至っていない．

実装した計測ツールは, 以下の条件を満たすアプリケーションを適用対象とする．

- Struts フレームワーク [3] を採用している
- データベースのアクセスに JDBC [5] を利用する
- 開発言語は Java である

6. 適用事例

6.1 概　　要

この適用事例では, 以下の仮説 H1〜H4 を検証する.

H1 システム全体の FP に関して, ツールの計測結果と CFPS(注1)の計測結果が似た値を示す,

H2 各 BFC(注2)の FP に関して, ツールの計測結果と CFPS の計測結果が似た値を示す,

H3 同一仕様に基づいて作成された複数のコードからは, ほぼ同じ結果が計測される,

H4 改良手法によって計測精度が向上する.

これらの検証を行うため, 7 つのグループが同一仕様を基にして独立に作成したシステムに対してツールを適用し, そのシステムの仕様書から熟練者が計測した結果との比較と (H1, H2). 各グループ間での比較を行った (H3). また, 既存手法による計測を行い, CFPS の計測結果と比較した (H4).

6.2 適用対象

適用対象は, 図書管理システムという, Struts と JDBC を用いて開発された Web アプリケーションである. 図書管理システムは, 学生 6 人で構成される 7 グループによって, 同一仕様を基にして開発された. また図書管理システムの FP は, CFPS によって仕様書から計測されている.

6.3 結　　果

表 1 は, CFPS が図書管理システムの FP を仕様書を基にして計測した結果である. 11 種類の TF と 4 種類の DF が計測され, システム全体の FP は 68 となった.

表 2 には, 7 グループが実装したソースコードからツールで計測した結果を示す. また表 3 には, CFPS 計測に対するツール計測の誤差 (MRE) を示す. MRE(Magnitude of Relative Error) は相対誤差の絶対値を意味し, “計測値 A に対する計測値 B の MRE” は次のように計測される

$$MRE = \frac{|A - B|}{A} \tag{1}$$

表 4 には, ツールが計測した総 FP の各グループ間での誤差を示す. 誤差は, グループ 1, 3, 4, 5, 6(総 FP : 69) とグループ 2(総 FP : 81) の間で最大 (17.4%) になった. また, 全組み合わせ (42 通り) の平均誤差は 5.4%となった.

表 5, 表 6 には, 既存手法の計測結果と, CFPS 計測に対する総 FP 計測誤差を示す. 7 グループの平均誤差は 4.4%となった.

6.4 考　　察

総 FP の平均誤差（MRE の平均値）は, 4.6%となった (表 3). 通常の計測, つまり仕様書からの計測においても 12%の誤差が発生する [6] ことを考えると, 4.6%の誤差は十分に実用的であるといえる. しかし, 各 BFC の FP に関する誤差は十分とは言えない（EI の平均誤差：24%, EO の平均誤差：100%）.

このことから, 仮説 H1 は実証されたが, 仮説 H2 は実証できなかったといえる. つまり, システム全体の FP は高い精度で計測できるが, その構成要素である BFC の FP に関する精度は十分でなかったということである. これは, 機能の抽出は成功しているが, 機能の分類に失敗していることが原因となって

表 1　CFPS が計測した FP

TFP(機能数)				DFP(機能数)			総 FP
EI	EQ	EO	計	ILF	EIF	計	(機能数)
17(5)	19(5)	4(1)	40(11)	28(4)	0(0)	28(4)	68(15)

表 2　ツールが計測した FP

group	TFP(機能数)				DFP(機能数)			総 FP
	EI	EQ	EO	計	ILF	EIF	計	(機能数)
1	19(6)	22(6)	0(0)	41(12)	28(4)	0(0)	28(4)	69(16)
2	31(6)	22(6)	0(0)	53(12)	28(4)	0(0)	28(4)	81(16)
3	19(6)	22(6)	0(0)	41(12)	28(4)	0(0)	28(4)	69(16)
4	19(6)	22(6)	0(0)	41(12)	28(4)	0(0)	28(4)	69(16)
5	19(6)	22(6)	0(0)	41(12)	28(4)	0(0)	28(4)	69(16)
6	19(6)	22(6)	0(0)	41(12)	28(4)	0(0)	28(4)	69(16)
7	22(6)	22(6)	0(0)	44(12)	28(4)	0(0)	28(4)	72(16)

表 3　CFPS 計測に対するツールの計測誤差 (MRE)

group	TFP				DFP			総 FP
	EI	EQ	EO	計	ILF	EIF	計	
1	0.118	0.158	1.00	0.025	0	N/A	0	0.015
2	0.823	0.158	1.00	0.325	0	N/A	0	0.191
3	0.118	0.158	1.00	0.025	0	N/A	0	0.015
4	0.118	0.158	1.00	0.025	0	N/A	0	0.015
5	0.118	0.158	1.00	0.025	0	N/A	0	0.015
6	0.118	0.158	1.00	0.025	0	N/A	0	0.015
7	0.294	0.158	1.00	0.100	0	N/A	0	0.059
最大値	0.823	0.158	1.00	0.325	0	N/A	0	0.191
最小値	0.118	0.158	1.00	0.025	0	N/A	0	0.015
平均値	**0.244**	0.158	**1.00**	0.079	0	N/A	0	**0.046**

表 4　各グループ間の総 FP 計測誤差 (MRE)

最大値	最小値	平均値
0.174	0	**0.054**

表 5　既存手法が計測した FP

group	TFP(機能数)				DFP(機能数)			総 FP
	EI	EQ	EO	計	ILF	EIF	計	(機能数)
1	16(6)	22(5)	0(0)	38(11)	28(4)	0(0)	28(4)	66(15)
2	28(6)	22(5)	0(0)	50(11)	28(4)	0(0)	28(4)	78(15)
3	16(6)	22(5)	0(0)	38(11)	28(4)	0(0)	28(4)	66(15)
4	16(6)	22(5)	0(0)	38(11)	28(4)	0(0)	28(4)	66(15)
5	16(6)	22(5)	0(0)	38(11)	28(4)	0(0)	28(4)	66(15)
6	16(6)	22(5)	0(0)	38(11)	28(4)	0(0)	28(4)	66(15)
7	19(6)	22(5)	0(0)	41(11)	28(4)	0(0)	28(4)	69(15)

表 6　CFPS 計測に対する既存手法の総 FP 計測誤差 (MRE)

最大値	最小値	平均値
0.147	0.015	**0.044**

(注1)：Certified Function Point Specialist : IFPUG によって認定される FP 計測資格保持者

(注2)：Base Functional Component : 機能構成要素. IFPUG 法では, EI, EO, EQ, ILF, EIF が相当する.

いる. 総 FP だけではなく, BFC の FP も重要である [7], [8] ことから, 機能分類の失敗を見逃すことは出来ない.

仮説 H3 に関しては, 同一仕様から独立に作られたシステムに対する計測値の平均誤差 (5.4%) が実用的な誤差の範囲に収まっていることから, 実証されたと考えられる.

また, 改良手法の有効性に関しては, 総 FP の平均誤差が改良前と変わらないため (改良前：4.4%, 改良後：4.6%), この適用事例では示すことが出来なかった. しかし, 機能抽出に関して改良後の判定の方が妥当であるという CFPS の意見を得ることができた (詳細は後述). TF 抽出改良手法に関しては今後, 適用事例を重ねた上で検討する必要がある.

以降は, 適用事例の中で観測された, 今後の課題となりうる問題について議論する.

ログイン機能と新規ユーザ登録機能の判定

図書管理システムには, ログイン機能と新規ユーザ登録機能という 2 つの機能が含まれる. この 2 つの機能は, 同じ画面遷移で実行される. 具体的には, 「新規ユーザ登録」というチェックボックスに対するチェックの有無で, どちらの機能が実行されるか否かが決まる.

これらの機能に関する判定は, 「CFPS による判定」と「既存手法による判定」, 「改良手法による判定」のそれぞれで異なる. CFPS は EO と判定し, 既存手法は EI と判定し, 改良手法では EI + EQ と判定している (図 4).

CFPS は, 新規ユーザ登録機能をログイン機能の副機能として扱っている. そして, ログイン機能が出力する「ユーザがログイン可能か否か」という情報が派生データであるという判断から, EO と分類している. 一方, 既存手法ではこれらの機能が同一画面遷移上で起こることから, まとめて一つの機能として抽出し, 新規ユーザ登録がデータベースに対して更新を行うため, EI と分類している. そして改良手法では, これらの機能が同時に実行されることがないため別の機能として抽出し, ログイン機能は算術演算を行わない出力機能なので EQ, 新規ユーザ登録は入力を行うので EI と分類している.

この結果を CFPS と共に議論したところ, これらの機能に対する判定は意見の分かれるところであるが, 改良手法による判定に関しては妥当なものであると言える, という意見を得た.

例外処理に使用される SQL

グループ 2, グループ 7 では, 他のグループに比べて FP が多く計測されている. これは, ある機能を実行した際に起こる例外処理に SQL を使用していることが原因となっている. これらの余分な SQL は, TF の複雑度を決定する DET や FTR の値を増加させ, 最終的に FP の増加につながっている. 例外処理はユーザに認識される機能ではないため, 通常の機能とは別に考えなければならない.

改良手法では SQL Tree から分岐構造が得られるため, これを適切に解析することで例外処理時に呼ばれた SQL は除去可能であると考えている.

7. ま と め

本研究では, これまで提案されてきた Web アプリケーション

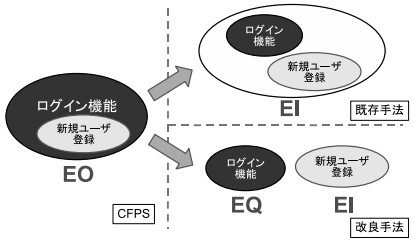

図 4　ログイン機能と新規ユーザ登録機能の判定

からの FP 自動計測手法に対する改良手法の提案と, ツールの実装を行った. そして, 実装したツールを Web アプリケーションに適用することで, 提案手法がシステム全体の FP に関して精度の高い結果が得られることと, 実装による違いをある程度吸収できることを確認した.

今後の課題としては, 本稿で提案したトランザクションファンクション分類手法の改良案をより詳細化してツールに実装することと, より多くの適用事例を行うことが挙げられる. 特に, 多くの適用事例を行うことは, 手法の詳細化のためにも重要である. まずは, 適用可能なフレームワークの拡大を行うことで, 様々な環境で開発されたアプリケーションに対して実験を行えるようにする予定である.

謝辞　本研究は一部, 文部科学省「次世代 IT 基盤構築のための研究開発」(研究開発領域名：ソフトウェア構築状況の可視化技術の開発普及) の委託に基づいて行われた. また, 日本学術振興会科学研究費補助金基盤研究 (C)(課題番号:20500033) の助成を得た. また, 本研究にあたってご協力頂いた株式会社 日立システムアンドサービスの英 繁雄様, 芝元 俊久様, 清水 健一様に感謝致します.

文　献

[1] A. J. Albrecht: “Function point analysis”, Encyclopedia of Software Engineering, **1**, pp. 518–524 (1994).
[2] International Function Point Users Groupg: “Function Point Counting Practices Manual Release 4.2” (2004).
[3] “The apache software foundation”, http://struts.apache.org/.
[4] 赤池, 楠本, 英, 芝元：“ソースコードからのファンクションポイント計測とその適用”, 電子情報通信学会技術研究報告, **107**, 392, pp. 97–102 (2007).
[5] “Sun developer network”, `http://java.sun.com/javase/technologies/database/`.
[6] B. A. Kitchenham: “The problem with function points”, IEEE Software, **14**, 2, pp. 29–31 (1997).
[7] A. Abran, B. Gil and E. Lefebvre: “Estimation models based on functional profiles”, International Workshop on Software Measurement, pp. 195–211 (2004).
[8] G. Gencel and L. Buglione: “Do different functionality types affect the relationship between software functional size and effort”, International Workshop on Software Measurement (2007).
[9] Common Software Measurement International Consortium: “COSMIC measurement manual”, 3.0 edition (2007).
[10] S. Kusumoto, M. Imagawa, K. Inoue, S. Morimoto, K. Matsusita and M. Tsuda: “Function point measurement from java programs”, Proc. of the 24th International Conference on Software Engineering, Orland, pp. 576–582 (2002).
[11] 柏本, 楠本, 井上, 鈴木, 湯浦, 津田：“イベントトレース図に基づく要求仕様書からのファンクションポイント計測手法”, 情報処理学会論文誌, **41**, 6, pp. 1895–1904.