修士学位論文

題目

静的解析による Web アプリケーションからの
ファンクションポイント自動計測手法の改善とその実験的評価

指導教員
楠本 真二 教授

報告者
枝川 拓人

平成 21 年 2 月 9 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻

平成 20 年度 修士学位論文

静的解析による Web アプリケーションからの
ファンクションポイント自動計測手法の改善とその実験的評価

枝川 拓人

内容梗概

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し, 開発現場で FP 法を効果的に利用する上で有効である. 私の所属する研究グループではこれまでに, 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.

本研究では, 既存手法において問題となっていた入出力機能の抽出と分類に関する改良手法を提案した. 具体的には, 条件分岐を考慮することで一つの画面遷移から複数の機能が抽出できるようにし, より詳細な分類定義に対応することとファジィ分類を取り入れることで分類精度の向上を図った. また, 既存手法の問題点である適用対象の少なさを改善するため, 手法を新たなフレームワークに対応させることで適用範囲を拡大した. そして, 改良手法を既存手法において対象とした Web アプリケーションと, 適用範囲の拡大によって適用可能になった Web アプリケーションに対して適用し, 手法の有効性について検証した. その結果, 抽出手法の改良の効果は確認できなかったものの, 分類手法の改良によって分類精度が向上することを確認した.

主な用語

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

目 次

1　まえがき

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手動で計測する場合, (1) 仕様書が残っていない場合計測できない, (2) 開発の中で起こる仕様変更が仕様書に反映されていない場合正確な値が得られない, (3) 過去プロジェクトのデータの準備と計測のためのコストが大きい, 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. この手法では, 画面遷移とデータベース上のテーブルを機能として抽出することで FP を計測する.

本稿では, 既存手法の問題点となっていた入出力機能の抽出と分類に関する改良手法を提案する. 具体的には, 条件分岐を考慮することで一つの画面遷移から複数の機能が抽出できるようにし, より詳細な分類定義に対応することとファジィ分類を取り入れることで分類精度の向上を図る. また, 既存手法の問題点である適用対象の少なさを改善するため, 計測手法を新たなフレームワークに対応させることで適用範囲を拡大させる. そして, 改良手法を既存手法において対象とした Web アプリケーションと, 適用範囲の拡大によって適用可能になった Web アプリケーションに対して適用し, 手法の有効性について, (1) 抽出手法の改良によって抽出精度が向上するか, (2) 分類手法の改良によって分類精度が向上するか (3)2 つの改良手法が計測精度を向上させるか, の 3 つの観点から評価を行う.

以降, 2 章ではファンクションポイント法について説明を行う. 続いて 3 章では既存手法とその問題点について述べる. 4 章では既存手法に対して行った改善点について説明し, 5 章では改良手法の評価を行うための適用事例について報告する. 6 章では FP 自動計測手法に関連する研究について述べる. 最後に, 7 章でまとめと今後の課題について記す.

2　ファンクションポイント法

2.1　概要

ファンクションポイント法は，A.J.Albrecht によって 1979 年に提案された．その後，これをベースに様々な改良手法 [3][2][4] が提案されている．その中でも現在では IFPUG（International Function Point Users Group）法 [2] が主流技法となっている．ファンクションポイント法は，利用者要求のうちの機能要求仕様の大きさを定量的に計測する手法であり，求められる計測値は，機能量または機能規模と呼ばれる．また，機能量の単位は伝統的にファンクションポイントと呼ばれている．

機能量の計測では，計測対象ソフトウェアの機能のうち，画面や帳票，ファイルなどを通じた情報の入出力に着目し，それらを種類別に数え上げ，種類数を加重合計した値を機能量とする．ただし，実現方法の違いによる影響を除くために，画面などの数は物理設計結果ではなく論理設計レベルで数える．

こうして得られた機能量の規模尺度の長所は，

- 規則にしたがって計測される値（推定値でなく誰が計測しても同じ値が得られる）
- 機能仕様にだけ依存（開発環境や開発言語などの技術用件に左右されない）

という二点である．これに対して，従来からソフトウェアの規模尺度として使われてきた SLOC 尺度は，開発言語や開発環境が変わると値が変化する，プログラムが完成するまでは値が確定しない，などの点で，機能量尺度より劣っている．

本研究では，数多くのファンションポイント法の中から，主流技法となっている IFPUG 法を基にしてファンクションポイントを計測する．

2.2　IFPUG 法

本研究では, 数多くのファンクションポイント (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 法の計測手順は図 1 のようになる．

IFPUG 法では, 計測する機能をデータファンクション (DF) とトランザクションファンクション (TF) の 2 種類に分類する. 実際に分類する際は, さらに細かく (1) 内部論理ファイル (ILF)，(2) 外部インタフェースファイル (EIF)，(3) 外部入力 (EI)，(4) 外部出力 (EO)，(5) 外部照会 (EQ) の 5 種類に分類する (図 2 参照). これら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.

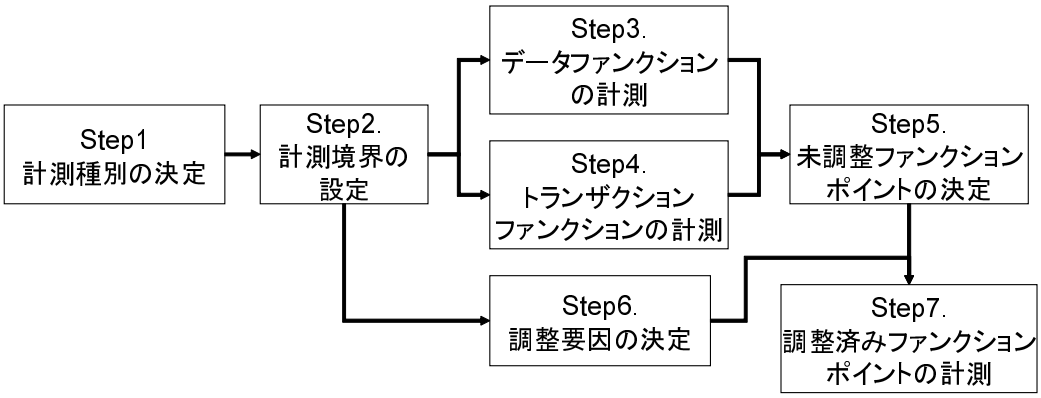

図 1: IFPUG 法の流れ

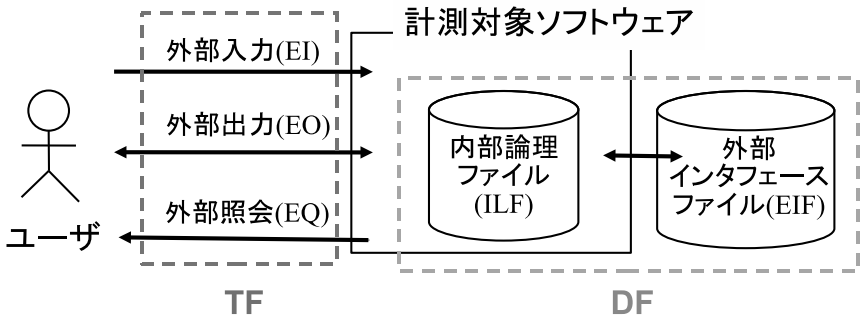

図 2: 機能の種類

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, 本研究では未調整 FP を対象として計測を行う.

以下に, IFPUG 法による FP 計測の主要プロセスである, Step3. データファンクションと Step4. 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.

Step3. 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, アプリケーション中にあり, 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, 内部論理ファイル (ILF) と外部インターフェイスファイル (EIF) の 2 種類に分類される. 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.

表 1: 複雑さの算出

データファンクション

RET \ DET	1〜19	20〜50	51〜
1	低	低	中
2〜5	低	中	高
6	中	高	高

外部入力

FTR \ DET	1〜5	6〜19	20〜
0〜1	低	低	中
2〜3	低	中	高
4〜	中	高	高

外部出力

FTR \ DET	1〜5	6〜19	20〜
0〜1	低	低	中
2〜3	低	中	高
4〜	中	高	高

外部照会

FTR \ DET	1〜4	5〜15	16〜
0〜1	低	低	中
2	低	中	高
3	中	高	高

表 2: 重みの算出

データファンクション

機能 \ 複雑さ	低	中	高
ILF	7	10	15
EIF	5	7	10

トランザクションファンクション

機能 \ 複雑さ	低	中	高
EI	3	4	6
EO	4	5	7
EQ	3	4	6

DF の複雑さは, データ項目数 (DET) と, レコード種類数 (RET) という 2 つのパラメータを表 1 に適用することで, 低・中・高の 3 段階に決定される. 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.

このように抽出し分類された DF は, 表 2 を用いて重みづけされ FP が得られる.

Step4. 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, 外部入力 (EI), 外部出力 (EO), 外部照会 (EQ) の 3 種類に分類される. 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, 以下の条件のいずれかを満たすものである.

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.

そして 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, データ項目数 (DET) と関連ファイル数 (FTR) の 2 つのパラメータによって, 表 1 に適用することで, 低・中・高の 3 段階に決定される. 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.

このように抽出し分類された TF は, 表 2 を用いて重みづけされ FP が得られる.

3　既存手法

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“ 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 3 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

1. SQL から DF を抽出する
2. DF を ILF, EIF に分類する,
3. DF の複雑度を判定する

Step3 : SQL から TF を識別する

1. SQL から TF を抽出する
2. TF を EI, EO, EQ に分類する,
3. TF の複雑度を判定する

Step4 : 未調整 FP を計算する

Step1 : SQL の抽出

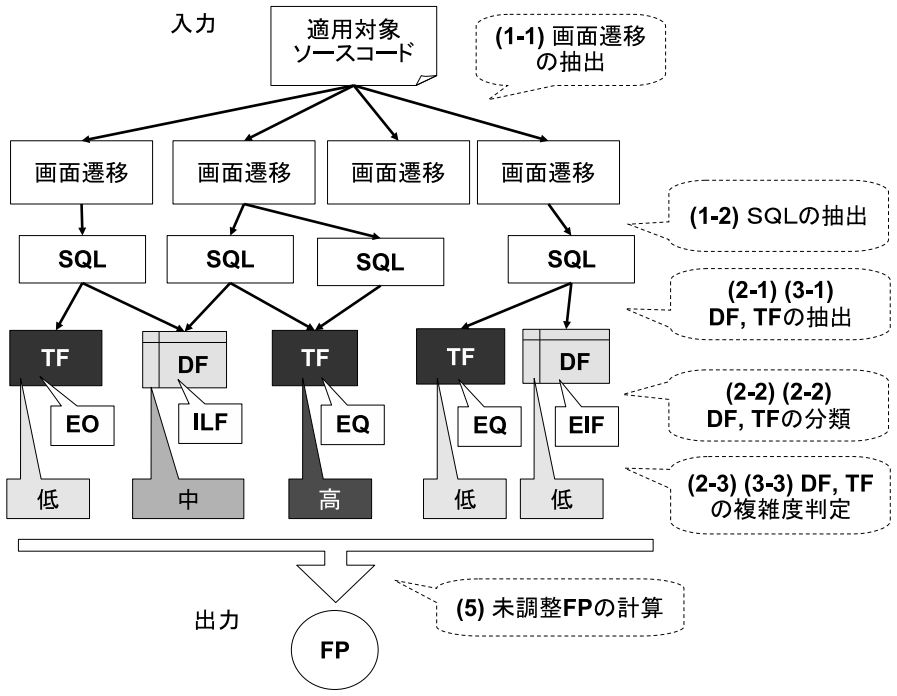

図 3: 計測手順

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.

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．

3.3 問題点

既存手法の問題点は大きく分けて 3 つある．

1. 機能の対応付けが不十分な場合がある
2. IFPUG による TF 分類定義を網羅していない
3. 対象とする Web アプリケーションの構成が限定されている

(1) 機能の対応付けが不十分な場合がある

既存手法では, DF がデータベース上のテーブル, TF が画面遷移時に実行される SQL の集合として実装されていることを仮定し, これらを抽出することで機能の抽出を行うことをキーアイデアとしている．しかし, この仮定は必ずしも正しいものではない．

まず, データベース上のテーブルでも DF になり得ない場合がある. 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. 既存手法ではこのようなテーブルはユーザが手動で除去する, としているが, そのように対処する場合, 自動計測の効果が薄れてしまう.

また, DF がデータベース上のテーブル以外で実装されている場合がある. あるシステムにおいては, 複数の画面遷移間で情報を共有する為の「セッション情報」にデータを保存するという形で, DF を実装している. このような場合, 既存手法では DF を抽出することは出来ない.

次に, TF が画面遷移時に実行される SQL の集合に対応するという仮定は, 複数の機能が一つの画面遷移に実装されている場合や, その逆に複数の画面遷移に一つの機能が実装されている場合を想定していない. このような機能の実装の仕方は一般的なものではないにせよ, 実際のシステムでも見受けられる実装方法である.

(2)IFPUG による TF 分類定義を網羅していない

既存手法が定義している分類手法は, IFPUG による分類定義のいくつかの部分を考慮していない.

まず, データベースを更新する SQL を EI として確定しているが, IFPUG の分類定義によれば, EO にも DF の状態を変更することが許されている.

また, データベースから取得したデータに対して算術演算がなされた時のみ, EO と判定しているが, 算術演算以外の加工がおこなわれる場合も, EO である可能性がある.

(3) 対象とする Web アプリケーションの構成が限定されている

手法が対象としているソフトウェアの限定条件が厳しすぎることが原因で, 実験対象が得づらく, 十分な検証が行えていない.

4　改善点

3.3 で挙げた問題点に対して, 手法の改良と適用範囲の拡大という二つの方向性で手法を改良する.

4.1　手法の改良

本研究では, 既存手法における TF 識別に関する二つの手法を改良した.

1. TF の抽出手法
2. 抽出した TF の分類手法

抽出手法の改良によって, 1 つの画面遷移に複数の画面遷移が実装されている場合に対応し, 分類手法の改良によって, IFPUG の分類定義により近い分類ができることを目指す.

4.1.1　条件分岐を考慮した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.

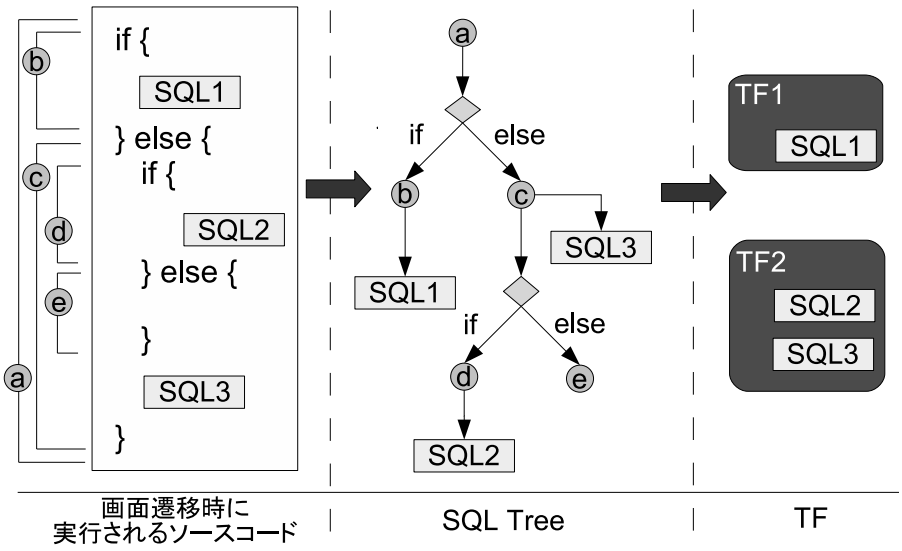

図 4: SQL Tree の構築と TF 抽出の例

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 4 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 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 4 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.

4.1.2 ファジィ分類を取り入れた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

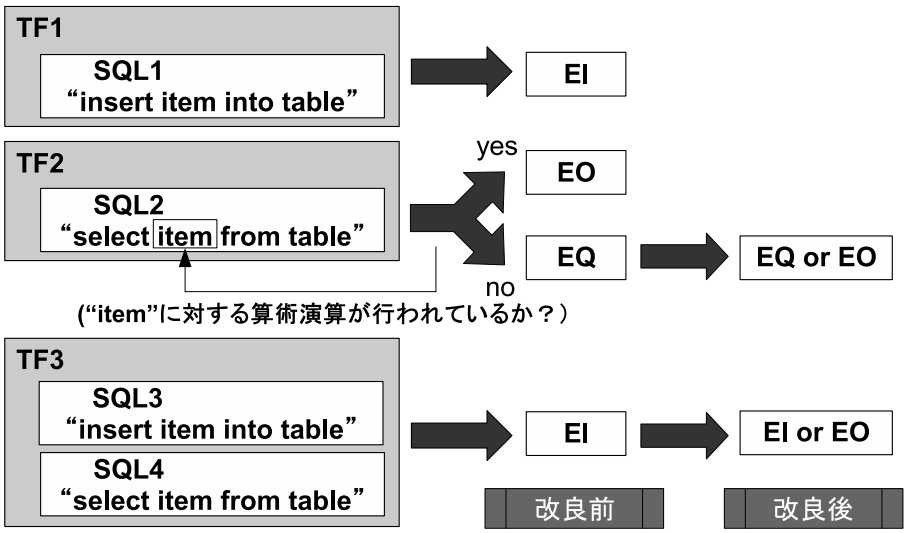

図 5: TF 分類の例

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 5 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が考えられる.

- 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については. 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.

しかし, これらのメトリクスをいかに上手く用いたところで,“ 100 %の確率でどちらかである ”という判定は不可能である. そこで、あいまいに分類する (ファジィ分類) という手法を採用した. つまり, メトリクスを用いて EI か EO を断定的に判定するのではなく, EI である確率と EO である確率を求めることで, 分類精度を向上させようというものである. この場合も, 多くの実験を重ねた上で, 確率決定アルゴリズムを決定する必要がある. 本研究では, 確率決定アルゴリズムを決定するだけの十分な予備実験が行えなかった為, 最も関連性が高いと考えられる「SQL の数」に着目して, 以下のアルゴリズムを採用した.

$$EI\text{である確率} = \frac{\text{入力に関連する } SQL \text{ の数}}{\text{出力に関連する } SQL \text{ の数} + \text{入力に関連する } SQL \text{ の数}} \tag{1}$$

$$EO\text{である確率} = \frac{\text{出力に関連する } SQL \text{ の数}}{\text{出力に関連する } SQL \text{ の数} + \text{入力に関連する } SQL \text{ の数}} \tag{2}$$

図 6 にファジィ分類の例を示す. この例では, 入力に関連する SQL が全体の 75%, 出力に関連する SQL が全体の 25%であるため, それらをそれぞれ EI である確率, EO である確率とする.

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.

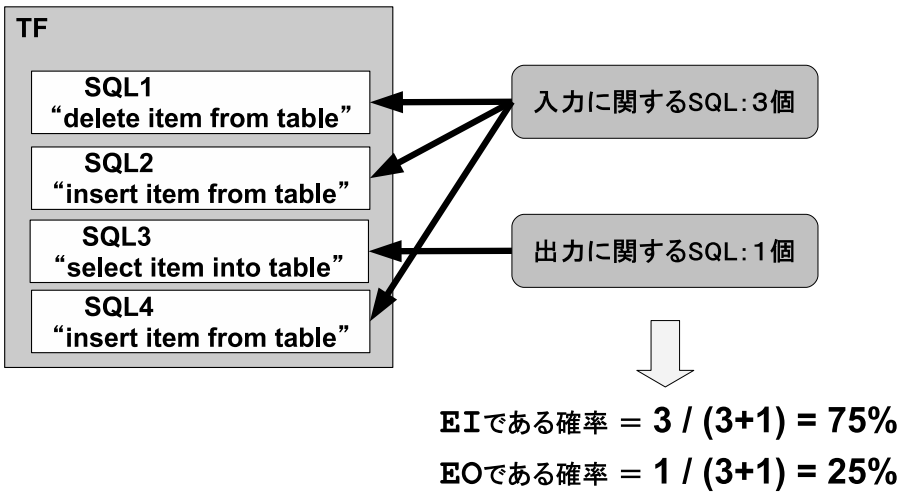

図 6: ファジィ分類の例

本研究では, SQL 文中に関数が使用される場合, 派生データを生成していると言えるのではないかと考え, 分類定義にその条件を加えた.

他に, 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. こちらに関しては, 現時点で検証は行えていない.

この派生データの生成の判定においてもファジィ分類を取り入れることで, 分類精度を向上させることができるのではないかと考えているが, 本研究の範囲では適切な確率決定アルゴリズムを考案できなかった為, 今後の課題とする.

4.2 適用範囲の拡大

既存手法における問題点の一つである, 適用範囲が狭いことで実験対象が得られにくい, という問題を解決するため, 適用範囲の拡大を行った.

具体的には, Spring フレームワークや iBATIS フレームワークを利用しているシステムを計測できるようにした. 何故これらのフレームワークに対応することにしたかというと, ある企業においてこれらのフレームワークを基に開発していこうという動きがあり, これらフレームワークに対応することで実験対象が得られやすいと考えた為である.

Spring フレームワークへの対応

Spring フレームワークでは, オブジェクトを動的に割り当てることが出来る. もしそのような実装がなされていた場合, 単純な静的解析では具体的に割り当てられるオブジェクトを

特定できない.

そこで, ソースコードの解析を行う前に, オブジェクトの割り当て設定が記述されているコンフィギュレーションファイルを解析することで, ソースコード解析中にオブジェクトの動的割り当てが発生した場合にマッピングをとれるようにした.

iBATIS フレームワークへの対応

iBATIS フレームワークでは, 呼び出す SQL を動的に割り当てることが出来る. その為, Spring フレームワークと同様に, 単純な静的解析では具体的に呼び出される SQL を特定できない.

そこで, ソースコードの解析を行う前に, SQL の割り当て設定が記述されているコンフィギュレーションファイルを解析することで, ソースコード解析中に SQL の動的割り当てが発生した場合にマッピングをとれるようにした.

5　適用事例

5.1　概要

この適用事例では, 表 3 の仮説 H1〜H3 を検証する.
これらの検証を行うため, 各適用対象に対して, 以下の 4 種の手法を適用する.

1. 既存手法
2. 既存手法に対して抽出手法のみを適用したもの
3. 既存手法に対して分類手法のみを適用したもの
4. 既存手法に対して両改良手法を適用したもの

そして, それらの結果と CFPS[1]が計測した結果を比較することで, 仮説が正しいかどうかを判断する.

5.2　適用対象

本研究で適用事例の対象としたのは, 図書管理システムと, 商品管理システムの 2 種類である.

図書管理システム

図書管理システムは, Struts, JDBC を用いて開発された Web アプリケーションである. このシステムは, ソフトウェア工学の授業において開発対象とされたもので, 学生 6 人で構成される 7 つのグループによって, 同一仕様を基にして開発された. 開発期間はおよそ 2 か月である. なお, 各グループは独立して開発を行っているが, 仕様はクラス図なども含み, 詳細にわたっているため, 開発された 7 つのプログラムは似通った内容になっている. また, このシステムは既存手法において適用事例に用いられたシステムである.

表 4 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. 本研究では TF の計測にのみ焦点を当てている為, TF の情報のみ掲載する. 11 種類の TF が計測され, その FP は 40 となった.

表 3: 仮説

H1	抽出手法の改良が抽出精度を向上させる
H2	分類手法の改良が分類精度を向上させる
H3	各改良手法が計測精度を向上させる

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

表 4: CFPS による計測結果 (図書管理システム)

FP(機能数)			
EI	EQ	EO	計
17(5)	19(5)	4(1)	40(11)

表 5: CFPS による計測結果 (商品管理システム)

FP(機能数)			
EI	EQ	EO	計
18(6)	21(7)	8(2)	47(15)

商品管理システム

商品管理システムは, Struts, Spring, iBATIS を用いて開発された Web アプリケーションである. このシステムは日立システムアンドサービスにおいて開発された, フレームワーク導入用のサンプルシステムである. なお, このシステムは本研究で行った適用範囲の拡大によって適用可能になったシステムである.

表 5 は, CFPS が商品管理システムの FP を仕様書を基にして計測した結果である. 15 種類の TF が計測され, その FP は 47 と計測された.

5.3 結果

ここでは, ツールによる計測結果と, 計測結果を仮説の検証の為に解析した結果を載せる.

5.3.1 計測結果

まず, 表 6 が商品管理システムに対する適用結果である. なお, この表の値は 7 グループの計測値の平均値である (数値は小数点第二桁で四捨五入したもの). グループごとの計測結果は付録 (表 13) に掲載している.

そして, 表 7 が商品管理システムに対する適用結果である. なお, 分類手法の改良を適用したときのみ, ファジィ分類によって数値に小数が発生する. その場合の数値は, 小数点第二で四捨五入したものを記述してある.

5.3.2 解析結果

仮説 1〜3 の検証するため, 計測結果に対して解析を行った.

この解析において, 誤差を評価する為に二つの指標を用いた. 解析結果を記述する前に, これらの指標について説明する. 使用した指標は, MRE(Magnitude of Relative Error) と

表 6: ツールによる計測結果 (図書管理システム)

適用した	FP(機能数)			
改良手法	EI	EQ	EO	計
なし	23.1(6.0)	19.0(5.0)	0(0)	42.1(11.0)
抽出手法	23.1(6.0)	22.0(6.0)	0(0)	45.1(12.0)
分類手法	16.6(4.7)	19.0(5.0)	6.2(1.3)	41.8(11.0)
両方	18.1(5.2)	22.0(6.0)	4.2(0.8)	44.3(12.0)

表 7: ツールによる計測結果 (商品管理システム)

適用した	FP(機能数)			
改良手法	EI	EQ	EO	計
なし	20(6)	26(8)	0(0)	46(14)
抽出手法	23(6)	32(10)	0(0)	55(16)
分類手法	9.8(3.0)	18.0(5.0)	21.1(6.0)	49.0(14.0)
両方	11.8(3.7)	24.0(8.0)	22.5(5.3)	58.3(17.0)

MER(Magnitude of Error Relative) の 2 種類である. “予測値 A に対する計測値 B の MRE, MER” は次のように計測される.

$$MRE = \frac{|A - B|}{A} \tag{3}$$

$$MER = \frac{|A - B|}{B} \tag{4}$$

これらは誤差を示す指標であるため, 小さい方が予測精度が高いということになる. また, 複数の MRE や MER の平均値を MMRE(Mean of MRE), MMER(Mean of MER) と記す.

なお, この章の表に記述した誤差は図書管理システム 7 つと商品管理システムそれぞれに対して解析した誤差の平均値である. それぞれの誤差に関する詳しい解析結果は付録に掲載してある.

以降, 解析結果を記述していく. 本研究で行った解析は, 以下の 4 種類である.

1. 抽出精度の解析
2. 分類精度の解析 (機能数の観点から)
3. 分類精度の解析 (FP の観点から)

表 8: 抽出精度

適用した改良手法	MMRE	MMER
なし	0.8%	0.9%
抽出手法	8.8%	8.1%
分類手法	0.8%	0.9%
両方	9.6%	8.8%

表 9: 分類精度 (機能数)

適用した改良手法	MMRE	MMER
なし	40.6%	9.1%
抽出手法	48.3%	18.1%
分類手法	21.0%	10.5%
両方	34.6%	16.5%

4. 計測精度の解析

まず, 抽出手法の改良効果を確かめる為, CFPS の計測結果に対する抽出機能数の誤差を解析した (仮説 1 の検証). 表 8 がその結果である. この表から, 抽出手法の改良によって抽出精度は下がっていることが分かる. この表では平均値のみを提示しているが, 個別の結果を見ても抽出精度は悪くなっていると言える (付録参照). よって, この適用事例では仮説 1 を確認することは出来なかったと言える.

次に, 分類手法の精度を確かめる為, 分類の結果得られた各機能数 (EI の数, EQ の数, EO の数) の CFPS 計測に対する誤差を解析し, 同様に各 FP の誤差を解析した (仮説 2 の検証). 表 9 が機能数の観点から誤差を解析した結果であり, 表 10 が FP の観点から誤差を解析した結果である. これらの表から, 分類手法の改良によって分類精度が向上していることが分かる. 機能数の観点からみた分類誤差 (MMER) のみ, 改良後の方が悪い結果になっているが, これに関しても改良前の手法において EO が観測されていない為, EO に関する MER が計測不能 (分母が 0 になる為) なことが原因となっており, 改良によって分類精度が悪くなったとは言えない. よって, この適用事例では仮説 2 は正しかったと言える.

そして, 両手法によって計測精度にどのような影響が出ているかを確かめる為, CFPS 計測に対する TF の FP の合計値の誤差を解析した (仮説 3 の検証). 表 11 がその結果である. この表から, 分類手法の改良のみ適用した場合の計測精度は少し上がっているが, その他は計測精度が下がっていることが分かる. 分類手法の改良による向上に関しても, 有意な差が出ているとは言い難いため, この適用事例では仮説 3 は確認できなかったと言える.

表 10: 分類精度 (FP)

適用した改良手法	MMRE	MMER
なし	44.5%	11.6%
抽出手法	50.3%	18.6%
分類手法	28.3%	10.1%
両方	32.6%	13.5%

表 11: 計測精度

適用した改良手法	MMRE	MMER
なし	5.0%	4.0%
抽出手法	13.4%	11.2%
分類手法	4.5%	3.8%
両方	12.4%	10.5%

最後に仮説の正否をまとめると, 表 12 のようになる.

5.4 考察

抽出手法改良の効果について

まず, なぜ抽出改良によって精度向上が得られなかったかについて考察する.

初めに, 今回の適用事例では改良前の手法による抽出精度が非常に高かった (MMRE:0.8%, MMER:0.9%) ということが言える. 本研究で抽出手法の改良を行ったのは, 一つの画面遷移に複数の機能が含まれている場合それを分割して抽出したい, という目的があったからであるが, 適用対象の 2 つのシステムには, そのような画面遷移が含まれていなかった. 既存の抽出手法で問題がなかった適用対象に対して, 改良手法を適用したところで精度向上が見込めないのは当然である. しかしながら, 精度が向上しないだけではなく悪化したということは, 改良手法が問題点を含んでいることは明らかである.

その問題点とは, 一つの機能として抽出すべき機能を複数の機能として抽出してしまっていることである. やや詳細な考察になるが, 上記の問題が起こった事例を 2 つ紹介する.

表 12: 仮説の正否

H1	抽出手法の改良が抽出精度を向上させる	×
H2	分類手法の改良が分類精度を向上させる	○
H3	各改良手法が計測精度を向上させる	×

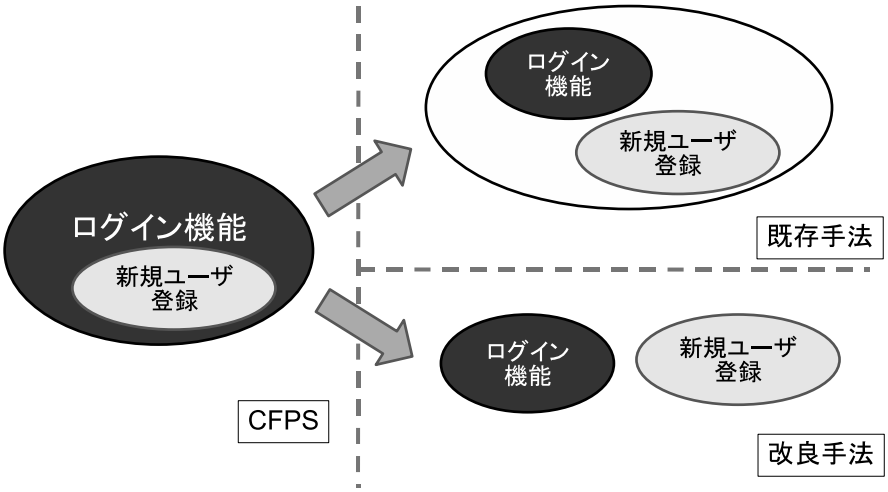

図 7: ログイン機能と新規ユーザ登録機能の抽出

ログイン機能と新規ユーザ登録機能

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.

これらの機能に対する抽出結果は,「CFPS による抽出結果」と「既存手法による抽出結果」,「改良手法による抽出結果」のそれぞれで異なる. (図 7).

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. つまり, これらは一つの機能として抽出されている. 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, これらの機能の扱い方はは意見の分かれるところであり, 改良手法による抽出結果に関しても妥当なものであると言える, という意見を得た.

実装方法による誤った分離処理

商品管理システムのあるメソッドでは, 引数でモードを指定することで, 実行する機能を選択するという実装方法がされていた.

もしこのメソッドを利用する画面遷移が, ユーザの入力値によってモードを指定する引数を制御していれば, 抽出改良手法が対象とする「一つの画面遷移に複数の機能が実装されている」パターンに当てはまる. しかし, 実際にこのメソッドを利用している幾つかの画面遷移では, モードを指定する引数は定数で与えられており, 利用する機能が固定されていた.

これに対して単純な静的解析を行った場合, 条件分岐によっていろいろな機能が実行されるように見える為, 抽出改良手法によって実際には一つの機能であるにも関わらず複数の機能に分離されてしまうことになる.

このような問題を解決する為には, 条件分岐の判定条件となる変数が, ユーザの入力値によって制御されているかどうかを追跡するという方法が考えられるが, 静的解析では完全な追跡は不可能である.

分類手法改良の効果について

5.3.2 での解析結果から, 平均的には分類精度が上昇していることは分かるが, 付録に解析した個々の結果をみるとわかるように, 商品管理システムにおいては分類精度は下がっている.

これは, 商品管理システムの EI の中に, SQL から情報を取得する命令 (出力に関する命令) を含むものが多く, EO が過剰に抽出されてしまったことが原因となっている.

対策としては, より良い確率決定アルゴリズムを模索するということが挙げられる. 現在のアルゴリズムは, 単純に出力命令と入力命令の比を, EO 確率と EI 確率に対応させているが, もともと IFPUG の定義においても, 入力を含む機能は EI である可能性が高く EO にも可能性として入力を許す, というニュアンスがあることから, 単純な比で考えるのではなく, いくらか EI の重みを高くして計測する必要があるかもしれない.

また, 現在は EI or EO の判定にのみファジィ分類を取り入れているが, EO or EQ も現状では確定的な分類が行えていない為, こちらにもファジィ分類を取り入れてみる価値はある.

これらの検討の為には, より多くの適用実験を行う必要がある.

計測精度への影響について

抽出手法の改良によって計測精度が下がったのは, 抽出機能数の精度が下がっている為当然と言える. しかし, 分類手法の改良によって分類精度自体は向上しているのに, 計測精度でみるとほとんど向上していない. これは, 分類の誤りがトータルの FP にあまり影響しないことを示している. 既存手法において分類精度があまり高くないにもかかわらず, 計測精度が非常に高いこともそのことを傍証している.

このことから, トータル FP の計測精度を重要視するのであれば, 分類精度よりも抽出精度が問題であることがわかる.

6 関連研究

この章では, FP 自動計測手法に関連する研究について述べる.

[5] では, WebML という設計言語から, FP 自動計測を行う手法が提案されている. こういった手法は設計段階で FP を計測することを念頭に開発されているため, 開発前に FP を予測するには都合がよいが, 開発終了後に FP を計測したいという立場からいえば, 有用性が低い場合がある. それは, もし開発段階でその設計言語で記述されていない場合, 限定された設計言語で設計書を書きなおすという非常に手間のかかる作業が必要になり, 開発終了後に低コストで FP を計測したいという要求に反する.

[6] では, データベースに関連する項目（table, relation, transaction, form, reports）を基に計算される, DBP というポイントを提案している. また, DBP を用いることで, MS-ACCESS に限定しているとは言え, 非常に精度の高い工数見積もりが可能であることを示している. われわれが提案している FP 計測方法は, 簡単にいえばテーブル (table) と, テーブルへのアクセス (transaction) を元に FP を計測するというものであるため, 手法として似通っている. しかしこの手法で計測される DBP はあくまで FP とは別のポイントであるため, 設計書から計測した FP との比較材料として用いることはできないのに対し, 我々の手法では FP を計測するため, FP との比較が可能である.

[7] では, 具体的な計測方法については触れられていないものの, ソースコードから FP を計測するための準備として, フォーマルな記述方法で FP 計測手順を表記し, ソースコードからの計測可能性を示唆している. 我々の手法は, IFPUG のルールを基に考案してはいるものの, 厳密に正しく IFPUG 法を守っているわけではなく, 実現可能な範囲で, できるだけ IFPUG 計測に近い値が出ることを重視した手法になっている. そのため, われわれの手法は厳密性という意味では問題があるかもしれないが, そのぶん実用性は高いと考えている.

これら関連研究と比較して, 本研究で改善対象としている手法は,

- 純粋に開発終了時のソースコードをターゲットとしていること
- 仕様書から計測した FP と比較可能な値を計測すること
- 実用的な手法であること

という点に, 新規性と有用性があると考えている.

7　あとがき

本研究では, これまで提案されてきたWebアプリケーションからのFP自動計測手法に対して改善を行い, その実験的評価を行った.

具体的には, トランザクションファンクションの抽出と分類に関する手法を改良し, 対象とするフレームワークを拡張することで適用範囲を拡大した. そして, 2つのWebアプリケーションを用いた適用実験を行い, 改良手法の効果を検証した. その結果, トランザクションファンクション抽出手法の改良に関しては抽出精度の向上が得られなかったが, 分類手法の改良によって分類精度が向上することを確認した. また, FPの計測精度に対して, 分類精度よりも抽出精度の方が影響するという知見を得た.

今後の課題としては以下のような事柄が挙げられる.

抽出改良手法の見直し

本研究で提案した抽出改良手法は, 適用事例によってその効果を確認することはできなかった. 適用対象によっては, 改良手法の方が精度の高い抽出が行えるという可能性はあるが, 5.4に提示したように, 現段階で改良手法が持ついくつかの問題点が明らかになっているので, それら問題点に対処しなければならない.

また, 本研究では一つの画面遷移から複数の機能を抽出することを目的としたが, その逆に複数の画面遷移に一つの機能が含まれる場合もあるので, その面での改良も必要である.

確率決定アルゴリズムの改良

分類改良手法については, 平均的には分類精度が向上したが, システムによってはむしろ分類精度が下がる結果になった. この原因として, 確率的分類の際に使用している確率決定アルゴリズムの検討が不十分であることが挙げられる. 他のアルゴリズムを考案し, より多くの適用実験を重ねた上で実用的な確率決定アルゴリズムを模索する必要がある.

適用事例の蓄積

本研究での適用事例では, 2つのWebアプリケーションに対して適用を行ったが, 正当な評価を行う為には, より多くの適用事例を重ねる必要がある. より多くの適用事例を行うことは, 分類手法における確率決定アルゴリズムを精錬する為にも有用である.

また, 今回適用対象とした2つのアプリケーションの内の1つは, 本研究で適用範囲を拡大することによって適用可能になったものであるが, 本研究の目的の一つである「適用範囲の拡大による実験対象の増加」を実現したとは言い難い.

4.2で述べたように, Spring, iBATISのフレームワークに対応することで実験対象を増やそうと試みたのは, ある企業においてその構成で開発を行っていこうという動きがあったためである. しかし現段階では, まだこれらのフレームワークを用いたシステムの開発がすす

んでいないこともあって多くのデータは得られなかった. 今後その企業での開発が進むにつれて, 実験対象が増えていくことが期待できる.

DF 識別に関する問題

本研究ではトランザクションファンクション識別に関する改良を行ったが, DF の識別に関する改良を行う必要がある.

具体的には, データベースのテーブルの中から DF ではないもの (ユーザに機能として認識されないもの) を取り除く手法と, データベースのテーブルとして DF が実装されていない場合に対応する手法を考案する必要がある.

謝辞

本研究において，常に適切な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻 楠本 真二 教授に深く感謝致します.

本研究において，常に適切な御指導および御助言を頂きました 同 岡野 浩三 准教授に深く感謝致します.

本研究において，常に適切な御指導および御助言を頂きました 同 肥後 芳樹 助教に深く感謝致します.

本研究において，常に適切な御指導および御助言を頂きました 同 柿元 健 特任助教に深く感謝致します.

本研究において，常に適切な御指導および御助言を頂きました 大阪大学大学院情報科学研究科情報ネットワーク学専攻 山口 弘純 准教授に深く感謝致します.

本研究において，評価実験に関してお手伝い頂いた株式会社 日立システムアンドサービス 芝本 俊久 氏に深く感謝致します.

本研究において，評価実験に関してお手伝い頂いた株式会社 日立システムアンドサービス 英 繁雄 氏に深く感謝致します.

本研究において，評価実験に関してお手伝い頂いた株式会社 日立システムアンドサービス 清水 健一 氏に深く感謝致します.

本研究において，評価実験に関してお手伝い頂いた日本ユニシス・ラーニング株式会社 毛利 幸雄 氏に深く感謝致します.

本研究において，評価実験に関してお手伝い頂いた 大阪大学大学院情報科学研究科コンピュータサイエンス専攻 三宅 達也 氏に深く感謝致します.

最後に，その他様々な御指導，御助言を頂いた大阪大学 大学院情報科学研究科コンピュータサイエンス専攻 楠本研究室の皆様に深く感謝致します.

参考文献

[1] A. J. Albrecht. Function point analysis. *Encyclopedia of Software Engineering*, Vol. 1, pp. 518–524, 1994.

[2] International Function Point Users Groupg. *Function Point Counting Practices Manual Release 4.2*, 2004.

[3] Common Software Measurement International Consortium. *COSMIC measurement manual*, 3.0 edition, 2007.

[4] http://www.nesma.nl.

[5] S. Ceri, P. Fraternali, M. Brambilla, A. Bongio, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2002. 978-1558608436.

[6] S. Abiad, R. Haraty, and N. Mansour. Software metrics for small database applications. In *ACM symposium on Applied computing*, Vol. 2, pp. 866–870, 2000.

[7] A. April, E. Merlo, and A. Abran. A reverse engineering approach to evaluate function point rules. In *Fourth Working Conference on Reverse Engineering*, 1997.

[8] G. Antoniol, C. Lokan, G. Caldiera, and R.Fiutem. A function point-like measure for object-oriented software. *Empirical Software Engineering*, Vol. 4, No. 3, pp. 263–287, sep 1999.

[9] Alain Abran, Blanca Gil, and Eric Lefebvre. Estimation models based on functional profiles. In *International Workshop on Software Measurement*, pp. 195–211, 2004.

[10] N. Bajaj, A. Tyagi, and R. Agarwal. Software estimation: a fuzzy approach. *ACM SIGSOFT Software Engineering Notes*, Vol. 31, No. 3, pp. 1–5, May 2006.

[11] J. Ceddia and M. Dick. Automating the estimation of project size from software design tools using modified function points. In *Proceedings of the Sixth Australasian Computing Education Conference*, Vol. 30, pp. 33–39, 2004.

[12] M. Collins and N. Duffy. Convolution kernels for natural language. In *Proceedings of The Neural Information Processing Systems 2001*, 2001.

[13] G. Costagliola, F. Ferrucci, G. Tortora, and G. Vitiello. Class point: An approach for the size estimation of object-oriented systems. *IEEE Transactions on Software Engineering*, Vol. 31, No. 1, pp. 52–74, 2005.

[14] P. Fraternali, M. Tisi, and A. Bongio. Automating function point analysis with model driven development. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, No. 18, October 2006.

[15] T. Fetcke, A. Abran, and T. Nguyen. Mapping the oo-jacobson approach into function point analysis. In *Technology of Object-Oriented Languages and Systems*, No. 23, pp. 192–202, 1997.

[16] Gigdem Gencel and Luigi Buglione. Do different functionality types affect the relationship between software functional size and effort. In *International Workshop on Software Measurement*, 2007.

[17] ibatis. `http://ibatis.apache.org/`.

[18] IFPUG. *It Measurement: Practical Advice from the Experts*. Addison-Wesley, 2002.

[19] Sun developer network. `http://java.sun.com/javase/technologies/database/`.

[20] C. Jones. *Applied Software Measurement -Assuring Productivity and Quality-*. McGraw-Hill, 1996. 978-0070328266.

[21] S. R. Kiebusch, B. Franczyk, and A. Speck. Metrics forsoftware system families. In *Proceedings of the 7th International ICSE-Workshop on Economics-Driven Software Engineering Research*, pp. 1–6, 2005.

[22] S. R. Kiebusch and B. Franczyk. Process family points versus (full) function points. In *International Conference on Software Engineering Proceedings of the 2006 international workshop on Economics driven software engineering research*, pp. pp15–18, Shanghai China, 2006.

[23] B. A. Kitchenham. The problem with function points. *IEEE Software*, Vol. 14, No. 2, pp. 29–31, 1997.

[24] Shinji Kusumoto, Katsuro Inoue, Takashi Kasimoto, Ayane Suzuki, Katsuhiko Yuura, and Micho Tsuda. Function point measurement for object-oriented requirements

specification. In *Proc. of International Computer Software and Applications Conference*, pp. 543–548, Taipei, Oct 2000.

[25] Shinji Kusumoto, Masahiro Imagawa, Katsuro Inoue, Shuuma Morimoto, Kouji Matsusita, and Michio Tsuda. Function point measurement from java programs. In *Proc. of the 24th International Conference on Software Engineering*, pp. 576–582, Orland, May 2002.

[26] G. C. Low and D. Ross Jeffery. Function points in the estimation and evaluation of the software process. *IEEE Transactions on Software Engineering*, Vol. 16, No. 1, pp. 64–71, jan 1990.

[27] Springframework. http://www.springframework.org/.

[28] The apache software foundation. http://struts.apache.org/.

[29] C. Symons. *Software Sizing and Estimating.* Wiley-Interscience, 1991. 978-0471929857.

[30] T. Uemura, S. Kusumoto, and K. Inoue. A function point measurement tool for uml design specifications. In *Proceedings of Sixth International Symposium on Software Metrics*, pp. 62–69, Florida, November 1999.

[31] Takuya Uemura, Shinji Kusumoto, and Katsuro Inoue. Function point analysis for design specifications based on the unified modeling language. *Journal of Software Maintenance and Evolution*, Vol. 13, No. 4, pp. 223–243, 2001.

[32] 赤池輝彦, 楠本真二, 英繁雄, 芝元俊久. 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, Vol. 107, No. 392, pp. 97–102, 2007.

[33] 枝川拓人, 赤池輝彦, 肥後芳樹, 楠本真二. 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. 電子情報通信学会技術報告, Vol. 108, No. 173, pp. 25–30, July 2008.

[34] 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, Vol. 41, No. 6, pp. 1895–1904, 2000.

付録

適用事例の章には載せられなかった詳細な実験データを掲載する.

まず, 図書管理システムに対して 4 つの手法 (改良なし, 抽出改良のみ, 分類改良のみ, 両改良) を適用した結果を, 表 13 に記す.

次に, 図書管理システム計測結果の CFPS 計測に対する誤差を記述する. まず FP に関する誤差データとして, 表 14 に MRE のデータを, 表 15 に MER のデータを載せる. 次に機能数に関する誤差データとして, 表 16 に MRE のデータを, 表 17 に MER のデータを載せる.

最後に, 商品管理システム計測結果の CFPS 計測に対する誤差を記述する. まず FP に関する誤差データとして, 表 18 に MRE のデータを, 表 19 に MER のデータを載せる. 次に機能数に関する誤差データとして, 表 20 に MRE のデータを, 表 21 に MER のデータを載せる.

なお, 商品管理システムに関する適用結果については, 適用事例の章で表 7 に示したので, 付録においては省略する.

表 13: 図書管理システムに対するツール適用結果

改良なし

group	FP(機能数)			
	EI	EQ	EO	計
1	21(6)	19(5)	0(0)	40(11)
2	33(6)	19(5)	0(0)	52(11)
3	21(6)	19(5)	0(0)	40(11)
4	21(6)	19(5)	0(0)	40(11)
5	21(6)	19(5)	0(0)	40(11)
6	21(6)	19(5)	0(0)	40(11)
7	24(6)	19(5)	0(0)	43(11)

分類改良のみ適用

group	FP(機能数)			
	EI	EQ	EO	計
1	16.5(5.0)	19(5)	4.5(1.0)	40.0(11.0)
2	17.5(3.2)	19(5)	13.7(2.8)	50.2(11.0)
3	16.5(5.0)	19(5)	4.5(1.0)	40.0(11.0)
4	16.5(5.0)	19(5)	4.5(1.0)	40.0(11.0)
5	16.5(5.0)	19(5)	4.5(1.0)	40.0(11.0)
6	16.5(5.0)	19(5)	4.5(1.0)	40.0(11.0)
7	16.5(4.5)	19(5)	7.0(1.5)	42.5(11.0)

抽出改良のみ適用

group	FP(機能数)			
	EI	EQ	EO	計
1	21(6)	22(6)	0(0)	43(12)
2	33(6)	22(6)	0(0)	55(12)
3	21(6)	22(6)	0(0)	43(12)
4	21(6)	22(6)	0(0)	43(12)
5	21(6)	22(6)	0(0)	43(12)
6	21(6)	22(6)	0(0)	43(12)
7	24(6)	22(6)	0(0)	46(12)

両改良適用

group	FP(機能数)			
	EI	EQ	EO	計
1	18.0(5.5)	22(6)	2.5(0.5)	42.5(12.0)
2	19.0(3.7)	22(6)	11.7(2.3)	52.7(12.0)
3	18.0(5.5)	22(6)	2.5(0.5)	42.5(12.0)
4	18.0(5.5)	22(6)	2.5(0.5)	42.5(12.0)
5	18.0(5.5)	22(6)	2.5(0.5)	42.5(12.0)
6	18.0(5.5)	22(6)	2.5(0.5)	42.5(12.0)
7	18.0(5.0)	22(6)	5.0(1.0)	45.0(12.0)

表 14: 図書管理システム計測結果の FP に関する誤差 (MRE)

改良なし

group	MRE			
	EI	EQ	EO	計
1	23.5%	0.0%	100.0%	0.0%
2	94.1%	0.0%	100.0%	30.0%
3	23.5%	0.0%	100.0%	0.0%
4	23.5%	0.0%	100.0%	0.0%
5	23.5%	0.0%	100.0%	0.0%
6	23.5%	0.0%	100.0%	0.0%
7	41.2%	0.0%	100.0%	7.5%
最大値	94.1%	0.0%	100.0%	30.0%
最小値	23.5%	0.0%	100.0%	0.0%
平均値	36.1%	0.0%	100.0%	5.4%

分類改良のみ適用

group	MRE			
	EI	EQ	EO	計
1	2.9%	0.0%	12.5%	0.0%
2	2.9%	0.0%	242.5%	25.5%
3	2.9%	0.0%	12.5%	0.0%
4	2.9%	0.0%	12.5%	0.0%
5	2.9%	0.0%	12.5%	0.0%
6	2.9%	0.0%	12.5%	0.0%
7	2.9%	0.0%	75.0%	6.3%
最大値	2.9%	0.0%	242.5%	25.5%
最小値	2.9%	0.0%	12.5%	0.0%
平均値	2.9%	0.0%	54.3%	4.5%

抽出改良のみ適用

group	MRE			
	EI	EQ	EO	計
1	23.5%	15.8%	100.0%	7.5%
2	94.1%	15.8%	100.0%	37.5%
3	23.5%	15.8%	100.0%	7.5%
4	23.5%	15.8%	100.0%	7.5%
5	23.5%	15.8%	100.0%	7.5%
6	23.5%	15.8%	100.0%	7.5%
7	41.2%	15.8%	100.0%	15.0%
最大値	94.1%	15.8%	100.0%	37.5%
最小値	23.5%	15.8%	100.0%	7.5%
平均値	36.1%	15.8%	100.0%	12.9%

両改良適用

group	MRE			
	EI	EQ	EO	計
1	5.9%	15.8%	37.5%	6.3%
2	11.8%	15.8%	192.5%	31.8%
3	5.9%	15.8%	37.5%	6.3%
4	5.9%	15.8%	37.5%	6.3%
5	5.9%	15.8%	37.5%	6.3%
6	5.9%	15.8%	37.5%	6.3%
7	5.9%	15.8%	25.0%	12.5%
最大値	11.8%	15.8%	192.5%	31.8%
最小値	5.9%	15.8%	25.0%	6.3%
平均値	6.7%	15.8%	57.9%	10.8%

表 15: 図書管理システム計測結果の FP に関する誤差 (MER)

改良なし

group	MER			
	EI	EQ	EO	計
1	19.0%	0.0%	N/A	0.0%
2	48.5%	0.0%	N/A	23.1%
3	19.0%	0.0%	N/A	0.0%
4	19.0%	0.0%	N/A	0.0%
5	19.0%	0.0%	N/A	0.0%
6	19.0%	0.0%	N/A	0.0%
7	29.2%	0.0%	N/A	7.0%
最大値	48.5%	0.0%	N/A	23.1%
最小値	19.0%	0.0%	N/A	0.0%
平均値	24.7%	0.0%	N/A	4.3%

分類改良のみ適用

group	MER			
	EI	EQ	EO	計
1	3.0%	0.0%	11.1%	0.0%
2	2.9%	0.0%	70.8%	20.3%
3	3.0%	0.0%	11.1%	0.0%
4	3.0%	0.0%	11.1%	0.0%
5	3.0%	0.0%	11.1%	0.0%
6	3.0%	0.0%	11.1%	0.0%
7	3.0%	0.0%	42.9%	5.9%
最大値	3.0%	0.0%	70.8%	20.3%
最小値	2.9%	0.0%	11.1%	0.0%
平均値	3.0%	0.0%	24.2%	3.7%

抽出改良のみ適用

group	MER			
	EI	EQ	EO	計
1	19.0%	13.6%	N/A	7.0%
2	48.5%	13.6%	N/A	27.3%
3	19.0%	13.6%	N/A	7.0%
4	19.0%	13.6%	N/A	7.0%
5	19.0%	13.6%	N/A	7.0%
6	19.0%	13.6%	N/A	7.0%
7	29.2%	13.6%	N/A	13.0%
最大値	48.5%	13.6%	N/A	27.3%
最小値	19.0%	13.6%	N/A	7.0%
平均値	24.7%	13.6%	N/A	10.7%

両改良適用

group	MER			
	EI	EQ	EO	計
1	5.6%	13.6%	60.0%	5.9%
2	10.5%	13.6%	65.8%	24.1%
3	5.6%	13.6%	60.0%	5.9%
4	5.6%	13.6%	60.0%	5.9%
5	5.6%	13.6%	60.0%	5.9%
6	5.6%	13.6%	60.0%	5.9%
7	5.6%	13.6%	20.0%	11.1%
最大値	10.5%	13.6%	65.8%	24.1%
最小値	5.6%	13.6%	20.0%	5.9%
平均値	6.3%	13.6%	55.1%	9.2%

表 16: 図書管理システム計測結果の機能数に関する誤差 (MRE)

改良なし

group	MRE			
	EI	EQ	EO	計
1	20.0%	0.0%	100.0%	0.0%
2	20.0%	0.0%	100.0%	0.0%
3	20.0%	0.0%	100.0%	0.0%
4	20.0%	0.0%	100.0%	0.0%
5	20.0%	0.0%	100.0%	0.0%
6	20.0%	0.0%	100.0%	0.0%
7	20.0%	0.0%	100.0%	0.0%
最大値	20.0%	0.0%	100.0%	0.0%
最小値	20.0%	0.0%	100.0%	0.0%
平均値	20.0%	0.0%	100.0%	0.0%

分類改良のみ適用

group	MRE			
	EI	EQ	EO	計
1	0.0%	0.0%	0.0%	0.0%
2	36.6%	0.0%	183.0%	0.0%
3	0.0%	0.0%	0.0%	0.0%
4	0.0%	0.0%	0.0%	0.0%
5	0.0%	0.0%	0.0%	0.0%
6	0.0%	0.0%	0.0%	0.0%
7	10.0%	0.0%	50.0%	0.0%
最大値	36.6%	0.0%	183.0%	0.0%
最小値	0.0%	0.0%	0.0%	0.0%
平均値	6.7%	0.0%	33.3%	0.0%

抽出改良のみ適用

group	MRE			
	EI	EQ	EO	計
1	20.0%	20.0%	100.0%	9.1%
2	20.0%	20.0%	100.0%	9.1%
3	20.0%	20.0%	100.0%	9.1%
4	20.0%	20.0%	100.0%	9.1%
5	20.0%	20.0%	100.0%	9.1%
6	20.0%	20.0%	100.0%	9.1%
7	20.0%	20.0%	100.0%	9.1%
最大値	20.0%	20.0%	100.0%	9.1%
最小値	20.0%	20.0%	100.0%	9.1%
平均値	20.0%	20.0%	100.0%	9.1%

両改良適用

group	MRE			
	EI	EQ	EO	計
1	10.0%	20.0%	50.0%	9.1%
2	26.6%	20.0%	133.0%	9.1%
3	10.0%	20.0%	50.0%	9.1%
4	10.0%	20.0%	50.0%	9.1%
5	10.0%	20.0%	50.0%	9.1%
6	10.0%	20.0%	50.0%	9.1%
7	0.0%	20.0%	0.0%	9.1%
最大値	26.6%	20.0%	133.0%	9.1%
最小値	0.0%	20.0%	0.0%	9.1%
平均値	10.9%	20.0%	54.7%	9.1%

表 17: 図書管理システム計測結果の機能数に関する誤差 (MER)

改良なし

group	MER			
	EI	EQ	EO	計
1	16.7%	0.0%	N/A	0.0%
2	16.7%	0.0%	N/A	0.0%
3	16.7%	0.0%	N/A	0.0%
4	16.7%	0.0%	N/A	0.0%
5	16.7%	0.0%	N/A	0.0%
6	16.7%	0.0%	N/A	0.0%
7	16.7%	0.0%	N/A	0.0%
最大値	16.7%	0.0%	N/A	0.0%
最小値	16.7%	0.0%	N/A	0.0%
平均値	16.7%	0.0%	N/A	0.0%

分類改良のみ適用

group	MER			
	EI	EQ	EO	計
1	0.0%	0.0%	0.0%	0.0%
2	57.7%	0.0%	64.7%	0.0%
3	0.0%	0.0%	0.0%	0.0%
4	0.0%	0.0%	0.0%	0.0%
5	0.0%	0.0%	0.0%	0.0%
6	0.0%	0.0%	0.0%	0.0%
7	11.1%	0.0%	33.3%	0.0%
最大値	57.7%	0.0%	64.7%	0.0%
最小値	0.0%	0.0%	0.0%	0.0%
平均値	9.8%	0.0%	14.0%	0.0%

抽出改良のみ適用

group	MER			
	EI	EQ	EO	計
1	16.7%	16.7%	N/A	8.3%
2	16.7%	16.7%	N/A	8.3%
3	16.7%	16.7%	N/A	8.3%
4	16.7%	16.7%	N/A	8.3%
5	16.7%	16.7%	N/A	8.3%
6	16.7%	16.7%	N/A	8.3%
7	16.7%	16.7%	N/A	8.3%
最大値	16.7%	16.7%	N/A	8.3%
最小値	16.7%	16.7%	N/A	8.3%
平均値	16.7%	16.7%	N/A	8.3%

両改良適用

group	MER			
	EI	EQ	EO	計
1	9.1%	16.7%	100.0%	8.3%
2	36.2%	16.7%	57.1%	8.3%
3	9.1%	16.7%	100.0%	8.3%
4	9.1%	16.7%	100.0%	8.3%
5	9.1%	16.7%	100.0%	8.3%
6	9.1%	16.7%	100.0%	8.3%
7	0.0%	16.7%	0.0%	8.3%
最大値	36.2%	16.7%	100.0%	8.3%
最小値	0.0%	16.7%	0.0%	8.3%
平均値	11.7%	16.7%	79.6%	8.3%

表 18: 商品管理システム計測結果の FP に関する誤差 (MRE)

改良なし

MRE			
EI	EQ	EO	計
11.1%	23.8%	100.0%	2.1%

分類改良のみ適用

MRE			
EI	EQ	EO	計
45.4%	14.3%	164.6%	4.3%

抽出改良のみ適用

MRE			
EI	EQ	EO	計
27.8%	52.4%	100.0%	17.0%

両改良適用

MRE			
EI	EQ	EO	計
34.4%	14.3%	181.3%	24.0%

表 19: 商品管理システム計測結果の FP に関する誤差 (MER)

改良なし

MER			
EI	EQ	EO	計
10.0%	19.2%	N/A	2.2%

分類改良のみ適用

MER			
EI	EQ	EO	計
83.1%	16.7%	62.2%	4.1%

抽出改良のみ適用

MER			
EI	EQ	EO	計
21.7%	34.4%	N/A	14.5%

両改良適用

MER			
EI	EQ	EO	計
52.5%	12.5%	64.4%	19.4%

表 20: 商品管理システム計測結果の機能数に関する誤差 (MRE)

改良なし

MRE			
EI	EQ	EO	計
0.0%	14.3%	100.0%	6.7%

分類改良のみ適用

MRE			
EI	EQ	EO	計
50.0%	28.6%	200.0%	6.7%

抽出改良のみ適用

MRE			
EI	EQ	EO	計
0.0%	42.9%	100.0%	6.7%

両改良適用

MRE			
EI	EQ	EO	計
38.8%	14.3%	166.5%	13.3%

表 21: 商品管理システム計測結果の機能数に関する誤差 (MER)

改良なし

MER			
EI	EQ	EO	計
0.0%	12.5%	N/A	7.1%

分類改良のみ適用

MER			
EI	EQ	EO	計
100.0%	40.0%	66.7%	7.1%

抽出改良のみ適用

MER			
EI	EQ	EO	計
0.0%	30.0%	N/A	6.3%

両改良適用

MER			
EI	EQ	EO	計
63.5%	12.5%	62.5%	11.8%