修士学位論文

題目

Java プログラムからのファンクションポイント計測手法

指導教官
井上 克郎 教授

報告者
今川 勝博

平成 13 年 2 月 14 日

大阪大学 大学院基礎工学研究科
情報数理系専攻 ソフトウェア科学分野

平成 12 年度 修士学位論文

Java プログラムからのファンクションポイント計測手法

今川 勝博

内容梗概

ソフトウェアの開発規模を見積もる手段として，ファンクションポイント法が用いられることが多くなっている．ファンクションポイント法は，一般的に要求仕様書や分析・設計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計測する手法である．ファンクションポイント法による見積もりを開発現場に導入する際，過去の開発において計測されたファンクションポイント値とその開発に要した開発工数や開発期間等の実績データを蓄積し，ファンクションポイント値とそれらの間に何らかの関係式を見出す必要がある．しかし，過去の開発において，要求仕様書や分析・設計書等の中間生成物が既に存在せず，最終生成物であるソースコードしか存在しないことも多い．そのような場合，ファンクションポイント計測が困難となる．

そこで，本研究では，オブジェクト指向開発されたプログラムからファンクションポイントを計測するための手法について検討し，その手法に基づいて Java ソースコードからのファンクションポイント計測ツールを開発した．そして，開発したツールを実際のシステムに対して適用し，ツールの計測したファンクションポイント値と，ファンクションポイント計測の熟練者が要求仕様書から計測したファンクションポイント値との比較を行った．比較の結果，ツールが計測した値は熟練者の測定した値に近い値であることが確認された．

主な用語

ファンクションポイント (Function point analysis)
見積もり (Estimation)
オブジェクト指向開発 (Object-oriented development)
Java (Java)

目 次

1 まえがき

ソフトウェアの大規模化・複雑化に伴い，高い品質を持ったソフトウェアを開発するためには，明確な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ことが重要になってきている．明確な開発計画を立てるためには，ソフトウェアの規模，投入する工数，開発期間，開発に使用される技術などを予測する必要があるが，中でも重要なものは開発工数と開発期間である [15]．通常，開発工数や開発規模を予測するのに，まずソフトウェアの規模を見積もり，これに基づいて開発工数と開発期間を予測する手法がとられている．

ソフトウェアの機能的な規模を見積もる手段の一つとして，1979 年に Albrecht によってファンクションポイント法 [1] が提案された．その後，これをベースに種々のファンクションポイント法が提案されている．現在，IFPUG 法 [6] と MarkII 法 [16] の 2 つの主流技法がある．ファンクションポイント法は，ソフトウェア開発の初期段階における成果物である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計測する手法である．従って，その値は開発環境や開発言語などの技術要件に左右されない一定の値になる．

一般に，ファンクションポイント法による見積もりを開発現場に導入する際には，過去の開発において計測されたファンクションポイント値とその開発に要した開発工数や開発期間等の実績データを蓄積し，ファンクションポイント値とそれらの間に何らかの関係式を見出す必要がある．蓄積されたデータ数が不十分な場合，関係式の正確性が低下し，開発規模の見積もりが不正確なものとなる．従って，過去の開発における成果物からファンクションポイントを計測しなければならない．しかし，ファンクションポイントを手作業で計測するためには教育や導入のためのコストが必要となる．更に，要求仕様書や設計仕様書等の中間成果物が存在せず，最終成果物であるソースコードしか存在しないことも多く，このような場合にはファンクションポイントの計測が困難となる．

そこで，本研究ではオブジェクト指向開発方法論に基づいて開発されたシステムを対象として，ソースコードからファンクションポイント値を計測するための手法について検討する．具体的には，ファンクションポイント法の主流技法の一つである IFPUG 法に基づいて，ソースコードからファンクションポイント値の計測を行うための計測ルールを提案する．更に，提案したルールをファンクションポイント計測ツールとして実装した上で，ある企業で Java 言語を用いて開発されたアプリケーションに対して適用し，その有効性の評価を行う．

以降，2 では，ファンクションポイント法，及び IFPUG 法について説明する．3 では，Java 言語 [5] を用いて開発されたシステムのソースコードからファンクションポイント値を計測するための手法について，4 では今回の提案手法に基づいて開発したファンクションポイント計測ツールについて説明する．5 では，実際に Java ソースコードから計測したファンク

ションポイント値についての評価と結果分析を行なう．最後に 6 で，まとめと今後の課題について述べる．

2　ファンクションポイント

2.1　ファンクションポイント法

ファンクションポイント法は，ユーザの要求から機能要求仕様の大きさを定量的に測定する手法で，A.J.Albrecht によって 1979 年に提案された．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，機能量の単位としては，伝統的にファンクションポイントという呼称が使われている．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，それぞれの複雑さによって重み付けを行ない加算した値を機能量とする．このようにして得られる機能量の規模尺度としての長所は，(1) 規則に従って計測される値であるため，誰が計測しても同じ値が得られる，(2) 開発環境や開発言語などの技術要件に左右されず，機能仕様にだけ依存する，という点が挙げられる．

現在，ファンクションポイント法は目的等に応じて様々な改良や変更が行なわれ，数十種類の計測方法がある [16][17]．本研究では，数多くのファンクションポイント法の中から，ファンクションポイント標準化の中心的組織である IFPUG が定めており，日本においても主流技法として用いられている IFPUG 法を用いてファンクションポイントの計測を行なう．

2.2　IFPUG 法

IFPUG 法 [6] は，Albrecht 版のファンクションポイント法に対して複雑さの評価の客観化やルールの精密化・適正化などの変更を行なったバージョンである．IFPUG 法は，図 1 に示す手順で計測する．

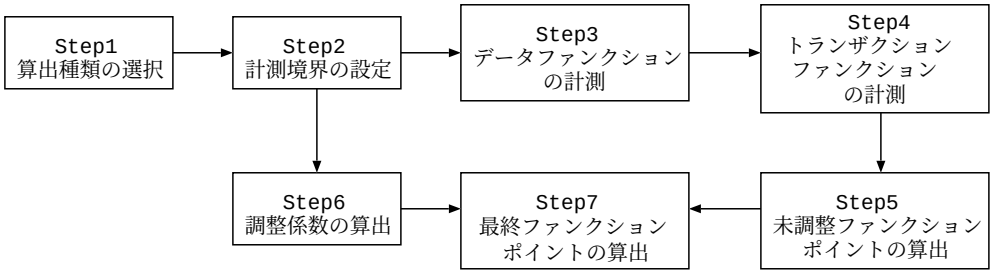

図 1: ファンクションポイント計測手順

Step1 算出種類の選択

算出種類を，

- アプリケーションファンクションポイント

アプリケーションソフトウェアの大きさを表すファンクションポイント.

- 新規開発プロジェクトファンクションポイント
  新規にアプリケーションを開発するプロジェクトの規模を知るために使用するファンクションポイント.

- 機能改良プロジェクトファンクションポイント
  すでに存在するアプリケーションを改良するプロジェクトの規模を知るために使用するファンクションポイント.

の 3 種類から選択する.

Step2 計測境界の設定

計測境界とは，ファンクションポイントを測定したい対象アプリケーション自体を境界の内部，他のアプリケーションおよびユーザを境界の外部になるように設定する境界のことである．境界の大きさはサブシステム程度を目安にし，一つの境界はユーザからみて機能面で閉じていなければならない.

Step3 データファンクションの計測

データファンクションとは，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意味でのデータのまとまりのことである．データファンクションにはファンクションタイプと複雑さの要素があり，それらを用いてデータファンクションのファンクションポイントが決定される．データファンクションには以下の二種類のファンクションタイプがある.

- 内部論理ファイル (Internal Logical File)
  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データファンクション.

- 外部インターフェースファイル (External Logical File)
  計測対象のアプリケーションによってデータが参照されるデータファンクション (データは更新されない).

次に，それぞれのデータファンクションをデータ項目数，レコード種類数という二つの要素によって低・中・高の三段階の複雑さに分類する.

- データ項目数 (Data Element Type,DET)
  データファンクションを構成する，ユーザが認識できる論理的なデータ項目の数.

- レコード種類数 (Record Element Type,RET)
  データファンクションの中に存在する，異なる意味合いを持つデータのまとまりの個数.

表 1: データファンクションの複雑さ

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

RET: レコード種類数　DET: データ項目数

更に，データファンクションの複雑さを決定するために表 1 を使用する．

Step4 トランザクションファンクションの計測

トランザクションファンクションとは，アプリケーションに対するデータの出入りを伴う処理のことである．トランザクションファンクションにもファンクションタイプと複雑さの要素があり，それらを用いてトランザクションファンクションのファンクションポイントが決定される．トランザクションファンクションには以下の三種類のファンクションタイプがある．

- 外部入力 (External Input,EI)
  計測境界外からのデータ入力によりデータファンクションが更新される処理.
- 外部出力 (External Output,EO)
  計測境界外へのデータ出力を含む処理のうち，データファンクションのデータを加工して出力する処理.
- 外部照会 (External Inquiry,EQ)
  計測境界とのデータ入出力により，データファンクションのデータを参照する処理.

次に，それぞれのデータファンクションをデータ項目数，関連ファイル数という二つの要素によって低・中・高の三段階の複雑さに分類する．

- データ項目数 (Data Element Type, DET)
  計測境界を出入りするデータ項目の個数.
- 関連ファイル数 (File Type Referenced, FTR)
  対象となるトランザクションファンクションの処理中にデータが更新または参照されるデータファンクションの個数.

複雑さを決定するためにファンクションタイプ毎に用意されている複雑さ決定表を使用する (表 2 参照).

表 2: トランザクションファンクションの複雑さ

外部入力の複雑さ

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

外部出力の複雑さ

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

FTR: 関連ファイル数　DET: データ項目数

Step5 未調整ファンクションポイントの算出

Step3，Step4 の結果をもとに，ファンクションタイプ・複雑さ別に表 3 を用いて重み付けを行ない，それらの値を合計した値が未調整ファンクションポイントとなる.

表 3: 未調整ファンクションポイント算出表

ファンクションタイプ \ 複雑さ	低	中	高	合計
内部論理ファイル	□ × 7 = □	□ ×10 = □	□ ×15 = □	
外部インターフェースファイル	□ × 5 = □	□ × 7 = □	□ ×10 = □	
外部入力	□ × 3 = □	□ × 4 = □	□ × 6 = □	
外部出力	□ × 4 = □	□ × 5 = □	□ × 7 = □	
外部照会	□ × 3 = □	□ × 4 = □	□ × 6 = □	
			未調整 FP	

Step6 調整係数の算出

未調整ファンクションポイントは「データのまとまり」と「データの出入り」のみに着目した値であり，性能，信頼性，ユーザーインタフェースなどについては考慮されていない. そこでシステム特性をファンクションポイントに反映させるために，表 4 に示すシステム特性の 14 項目を 6 段階で評価し，その結果から調整係数を算出する. 調整係数はファンクションポイント算出の際に，未調整ファンクションポイントを補正する役割がある.

Step7 最終ファンクションポイントの計測

未調整ファンクションポイントと調整係数を用いて最終ファンクションポイントを算出す

表 4: システム特性の 14 項目

1	データ通信機能	8	オンライン更新
2	分散データ処理	9	複雑な処理
3	性能条件	10	再利用性
4	高負荷構成	11	インストレーションの容易さ
5	トランザクション率	12	運用の容易さ
6	オンラインデータ入力	13	複雑サイト
7	エンドユーザーの効率	14	変更の容易さ

$$\text{調整係数} = 0.01 \times \text{影響度の合計} + 0.65$$

る．Step1 の算出種類によって算出方法が異なる．

- アプリケーション FP = 未調整 FP × 調整係数
- 新規開発 FP = (未調整 FP + 移行分未調整 FP) × 調整係数
- 機能改良 FP = (変更部分の新未調整 FP + 追加部分の未調整 FP + 移行分未調整 FP) × 新調整係数 + 削除部分の未調整 FP × 旧調整係数

2.3 実用面における課題

定量的にソフトウェアの機能量を計測することができるファンクションポイント法を用いることで，ソフトウェア開発プロジェクトに際して，開発工数や開発期間を見積もることが可能となる．

しかし，詳細な部分のファンクションポイントの計測には測定者の判断が必要になる．結果として，同一プロダクトに対してのファンクションポイントの計測であっても，計測する人間によって誤差が生じてしまうという問題点が指摘されている [14]．例えば，同じ組織内の人間が同じプロダクトに対して測定した場合は 12%，違う組織の人間が測定した場合は 30%以上の誤差が出るという報告もされている [12]．

また，ファンクションポイントを実際に応用して見積りを行うためには，過去の開発において計測されたファンクションポイント値とその開発に要した工数や期間に関するデータを蓄積し，その関係式を導き出さなければならない．そのためには関係式が得られるだけの十分な過去の開発に関するデータが必要となり，その計測のためのコストが現場への導入の妨げとなっている．更に，一般にファンクションポイントは要求仕様書や設計仕様書から計測

されるが，過去の開発における成果物として，最終成果物であるソースコードしか存在しないことも多く，そのような場合にはファンクションポイント値の計測が難しい．

そこで，以降ではソースコードからファンクションポイント値の計測を行なうための手法について検討する．

3 提案するファンクションポイント計測手法

ここでは，オブジェクト指向プログラミングで作成されたシステムからファンクションポイントを計測するために提案する手法について説明する．本提案手法では，IFPUG 法に基づいて計測を行なうため，中心となるのはシステムからの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二種類のファンクションを抽出する作業である．以下では，まずファンクションポイントの基本的な計測方針について述べ，次にファンクションの抽出方法と複雑さの決定方法，及び，今回試作したファンクションポイント計測ツールについて説明する．

3.1 基本方針

ファンクションの基本的な抽出方針として，計測対象となるシステムのすべての機能に対応するような入力データを，例えばユースケース [11] に対応させて作成する．作成された入力データに基づいて実際にシステムを動作させて，ファンクションポイント計測に必要な情報を得る方法をとる．

ファンクションポイント計測に必要な情報を，ソースコードの静的な解析結果のみからではなくシステムの実行過程から抽出する理由としては，主に次の 2 点がある．

- 実際に動作する機能に対するファンクションポイントを計測するため．つまり，ソースコード中に含まれるユーザに見えないコード等 (例えば，開発保守用のコード) は計測対象としない．
- 計測に必要な情報量を少なくするため．

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それぞれの基本的な抽出方針を以下に示す．

- データファンクション: システムを構成するクラスのいずれかをデータファンクションとする．IFPUG 法におけるデータファンクションは，システム内に存在する論理的なデータのまとまりのことを指すが，オブジェクト指向におけるクラスも同様の意味を持つと考える．
- トランザクションファンクション: データファンクションであるクラスで定義されているメソドが他のクラスにおいて呼び出されたとき，そのメソド呼び出しをトランザクションファンクションを構成する要素として抽出する．IFPUG 法におけるトランザクションファンクションは，データファンクションを更新・参照するようなシステム

への入出力を伴う処理のことであるが，データファンクションをクラスと考えたときに，クラスが保持する情報を更新・参照するメソド呼び出しがトランザクションファンクションとなる可能性があると考える．トランザクションファンクションを構成する要素を，実際に計測対象となるトランザクションファンクションとして識別するための方法については，3.3.1 で述べる．

以降では，ファンクションの識別方法と複雑さの決定方法の詳細について説明する．

3.2 データファンクションの計測

3.2.1 データファンクションの識別

計測対象システムを構成するクラスの中から，ユーザがデータファンクションをクラス単位で指定する．

次に，データファンクションを内部論理ファイルまたは外部インターフェースファイルのいずれかに分類する．分類方法は，データファンクションに指定されたクラスについて，システムの実行中に，引数が渡されるメソド呼び出しが一度でもある場合は，内部論理ファイルとする．つまり，メソド呼び出しの際に渡される引数によって，クラスの保持するデータが更新されると考える．そのようなメソド呼び出しが一度もない場合には，クラスの保持するデータは更新されることがないと考え，外部インターフェースファイルとする．

3.2.2 データファンクションの複雑さ

IFPUG 法では，データファンクションの複雑さを決定する要素として，DET と RET の二つがある．DET は，データファンクションを構成するデータ項目の個数のことを意味し，RET は一つのデータファンクション中に存在する異なる意味合いを持つデータのまとまりの個数を意味する．

本手法ではクラスをデータファンクションに対応させているため，DET はクラスで定義されているクラス変数の中で，int，char，boolean 等の型として宣言されている変数の個数とし，RET はクラス型として宣言されているクラス変数の個数とする．つまり，クラス型として宣言されているクラス変数は，ある意味合いを持ったデータのまとまりであると考える．

DET，RET の値によって，データファンクションの複雑さを低・中・高の三段階に分類し，その複雑さによって重み付を行ないファンクションポイントを算出する．その際に用いる分類表と算出表は IFPUG 法で用いられているものと同じものを使用する．

3.3 トランザクションファンクションの計測

3.3.1 トランザクションファンクションの識別

まず，データファンクションに指定されたクラスで定義されているメソドが他のクラスにおいて呼び出されたとき，それをトランザクションファンクションを構成する基本要素として抽出する．つまり，システムの実行中に，図 2 において太線矢印で示されるメソド呼び出しが出現すれば，データファンクションである classB の保持するデータが更新・参照されると考え，そのメソド呼び出しをトランザクションファンクションの基本要素として抽出する．

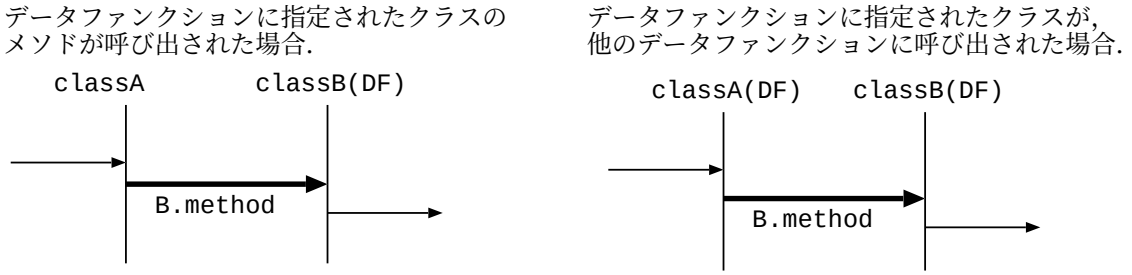

図 2: トランザクションファンクションの基本要素

次に，抽出されたメソド呼び出しについて，それらを 1 対 1 でトランザクションファンクションとして識別すると，要求仕様書上で一つのトランザクションファンクションとして計測された機能要件が，ソースコード上でも一つのメソド呼び出しで実装されていなければならないことになる．

そこで，ユーザからシステムへ何らかの入力が行なわれた時に，必ずメソドが呼び出されるようなクラス (例えば GUI クラスやサーブレットクラス) を境界クラスとして指定する．そして境界クラスに定義されているメソドが呼び出されてから，そのメソド呼び出しが終了するまでの間に，図 2 のルールに基づいて抽出されたメソド呼び出しが一度でも出現する場合は，その一連のメソド呼び出しを一つのトランザクションファンクションとして識別する (図 3 参照).

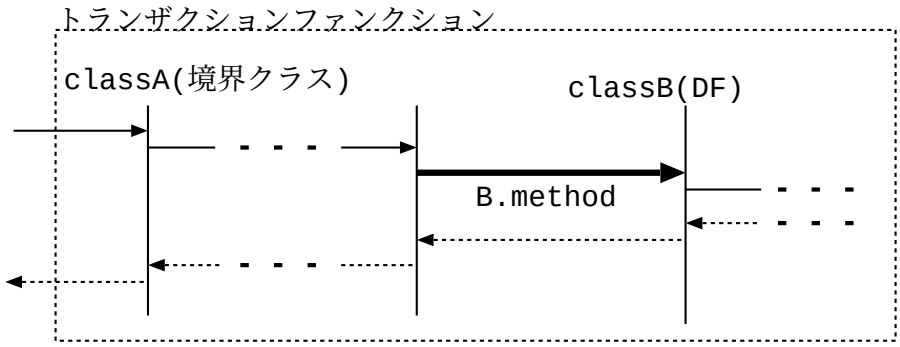

図 3: 境界クラスによるトランザクションファンクションの識別

以上のようにして識別したトランザクションファンクションについて，外部入力・外部出力・外部照会のいずれかに分類する．そのルールを図 4 に示す.

ルール 1 では，ユーザが定義したクラスが引数としてデータファンクションクラスに渡され，その引数によってデータファンクションクラスの保持するデータが更新されるとして外部入力とする.

ルール 2 では，DF クラスからの返り値が境界クラスに渡される前に，データファンクションに指定されていないクラス (中間クラス) によって何らかのデータの加工が行なわれ境界クラスへの返り値となると考える．IFPUG 法において，データファンクションの保持するデータが加工され計測境界外へと出力されるのは外部出力であるので，ルール 2 のように中間クラスが存在する場合は外部出力とする.

ルール 1，ルール 2 を共に満たさないようなトランザクションファンクションはデータファンクションの更新や計測境界外への出力データの加工が行なわれない処理として外部照会とする.

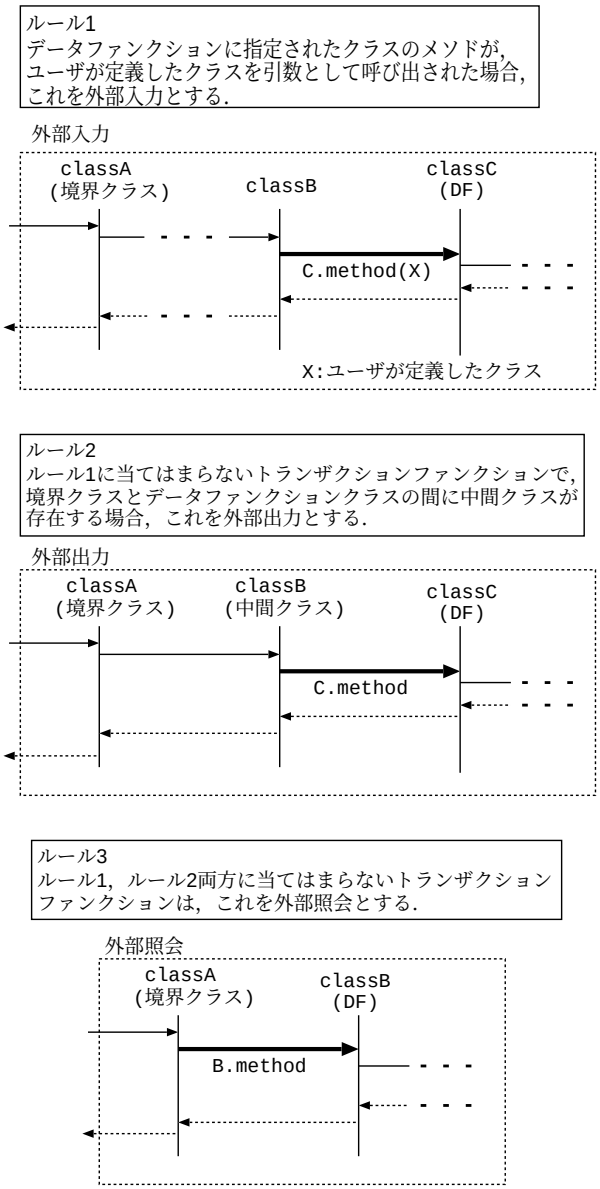

図 4: トランザクションファンクションの分類ルール

3.3.2 トランザクションファンクションの複雑さ

IFPUG 法では，トランザクションファンクションの複雑さを決定する要素として，DET と FTR の二つがある．DET は計測境界を出入りするデータ項目の個数のことを意味し，FTR は対象となるトランザクションファンクションの処理中にデータが更新または参照されるデータファンクションの個数を意味する．

本手法のトランザクションファンクション識別ルールにおいて，計測境界となるのは境界クラスである．従って，DET についてはトランザクションファンクションの種類別に以下のように定義できる．

- 外部入力の場合
  外部入力の処理中に境界クラスから呼び出されるメソドの引数の個数の合計．
- 外部出力の場合
  外部出力の処理中に境界クラスから呼び出されるメソドの返り値の個数の合計．返り値がクラスである場合にはそのクラスで宣言されているクラス変数の個数を用いる．
- 外部照会の場合
  外部出力の場合と同じ．

FTR はトランザクションファンクション中に出現するデータファンクションクラスの個数とする．同一のデータファンクションクラスが重複して出現する場合は，一度だけカウントする．

また，その他のルールとして，トランザクションファンクションの処理中に呼び出されるメソドとその引数の型・個数，呼び出される順序がすべて一致するトランザクションファンクションは同一トランザクションファンクションとして一度だけ計測する．

DET，FTR の値によって，トランザクションファンクションの複雑さを低・中・高の三段階に分類し，その複雑さによって重み付を行ないファンクションポイントを算出する．その際に用いる分類表と算出表は IFPUG 法で用いられているものと同じものを使用する．

4 計測ツール

4.1 ツールの概略

今回提案した手法に基づいて，Java ソースコードからファンクションポイント計測を行なうためのツールを開発した．ツールは Windows2000 上で Java を用いて実装している．

4.2 ツールの構成

図 5 に計測システムの構成を示す．開発したシステムは構文解析部，実行部，FP 計測部の三つの機能を持つ．

- 構文解析部
  計測対象となるシステムのソースコードを入力として，FP 算出に用いるための構文情報ファイルを出力する．構文情報ファイルの詳細は後述する．

- 実行部
  対象となるシステムのソースコードとユースケースを入力として，実際に計測対象システムを動作させ，その実行の過程から FP 算出に必要な情報を抽出し，実行ログファイルとして出力する．実行ログファイルについての詳細も後述する．

- FP 計測部
  構文情報ファイルと実行ログファイル，及びユーザによるデータファンクションクラスと境界クラスの指定を入力として FP を算出し出力する．

4.3 構文情報ファイル

構文情報ファイルの形式を表 5 に示す．構文情報ファイルに記述されている情報として，まずクラス名がある．次に，そのクラスで宣言されているクラス変数の名前，その変数の型，フラグ (その変数がデータファンクションの複雑さを決定する際に DET と見なされるのか，もしくは RET と見なされるのかを判別するために用いる) からなる「CV」の行がある．次に，そのクラスで宣言されているメソド名があり，そのメソドで宣言されている変数についても，クラス変数と同様の情報が「MV」の行に記述されている．表 5 は一つのクラスについての情報で，構文情報ファイルには全クラス分がこのような形式で記述されている．

4.4 実行ログファイル

実行ログファイルの例を表 6 に示す．実行ログファイルにおいて,「Begin」で始まる行はシステムの実行中にあるメソドが呼び出されたことを意味する．逆に,「End」で始まる行は

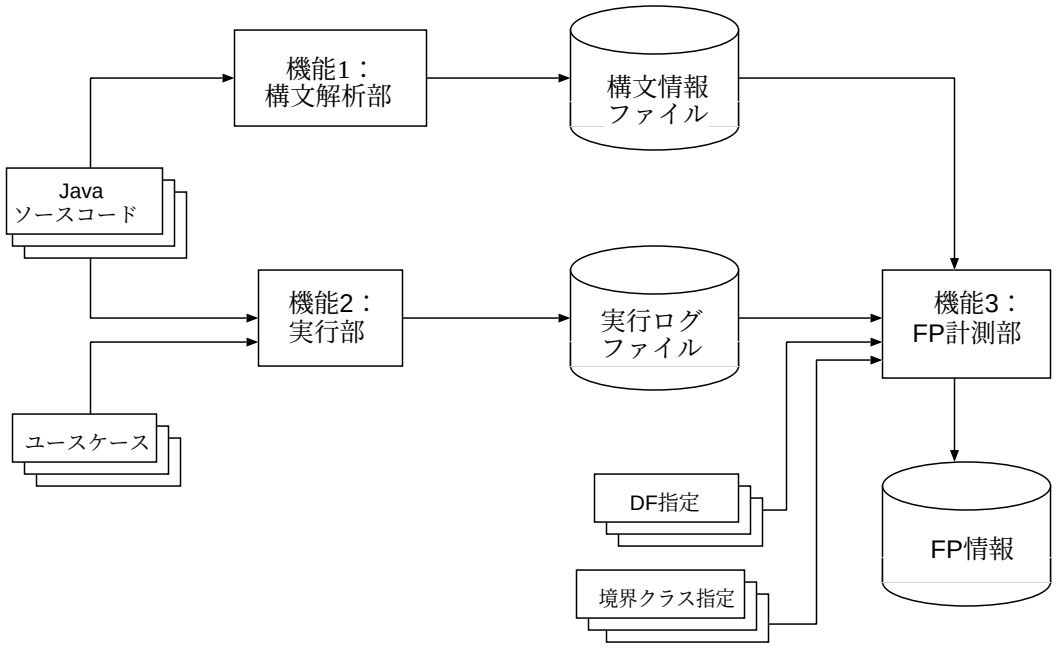

図 5: 計測システムの構成

あるメソドが終了したことを意味する.「Begin」「End」の後には「クラス名.メソド名.引数の個数」と続き，引数の個数に応じて引数の型が記述される．表 6 の例では，まず，Sakaya クラスの Sakaya メソドが引数一つで呼び出され，その引数は String 型である．次に Souko クラスの Souko メソドが引数二つで呼び出され，その引数は String 型と int 型である．最後に Haisou クラスの Haisou メソドが引数なしで呼び出され，呼び出されたメソドが再帰的に終了している．表 6 を図式化したものを図 6 に示す.

表 5: 構文情報ファイルの形式

C	クラス名		
CV	クラス変数名	型	DETflag
⋮	⋮	⋮	⋮
M	メソド名		
MV	メソド変数名	型	DETflag
⋮	⋮	⋮	⋮
M			
MV			

表 6: 実行ログファイルの例

Begin	Sakaya.Sakaya.1.String
Begin	Souko.Souko.2.String.int
Begin	Haisou.Haisou.0
End	Haisou.Haisou.0
End	Souko.Souko.2.String.int
End	Sakaya.Sakaya.1.String

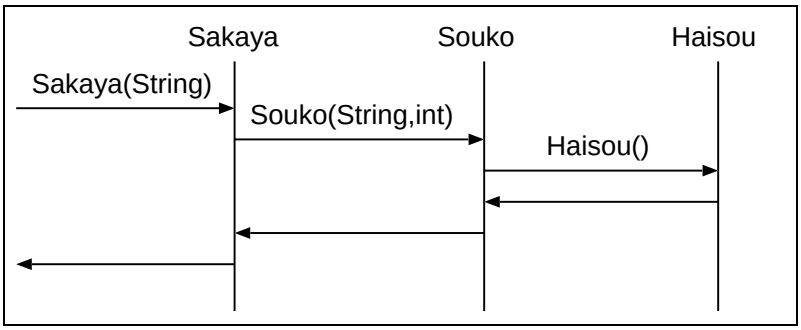

図 6: 表 6 の対応図

4.5 ツールの実行例

ここでは試作したファンクションポイント計測ツールについて，構文解析部・実行部・FP 計測部の機能別にツールの実行過程の画面を用いて説明する．

4.5.1 構文解析部

機能選択から構文解析を選択する (図 7 参照)．そして「start」ボタンを押すと [Input compile & execute command] ダイアログ (図 8) が表示される．このダイアログでは，

- Java プログラムが存在するフォルダ
  構文解析や実行の対象となるプログラムが存在するフォルダの指定を行なう．参照ボタンを押すと，[Select Directory] ダイアログ (図 9) が表示されるので，これを利用してフォルダ名を入力することができる．

- ログファイルの出力フォルダ
  構文解析部の出力ファイルである構文情報ファイルや実行部の出力ファイルである実行ログファイルの出力先フォルダの指定を行なう．

- コンパイルコマンド
  実行部を利用する際に計測対象となるプログラムのコンパイルを行なう．通常コンパ

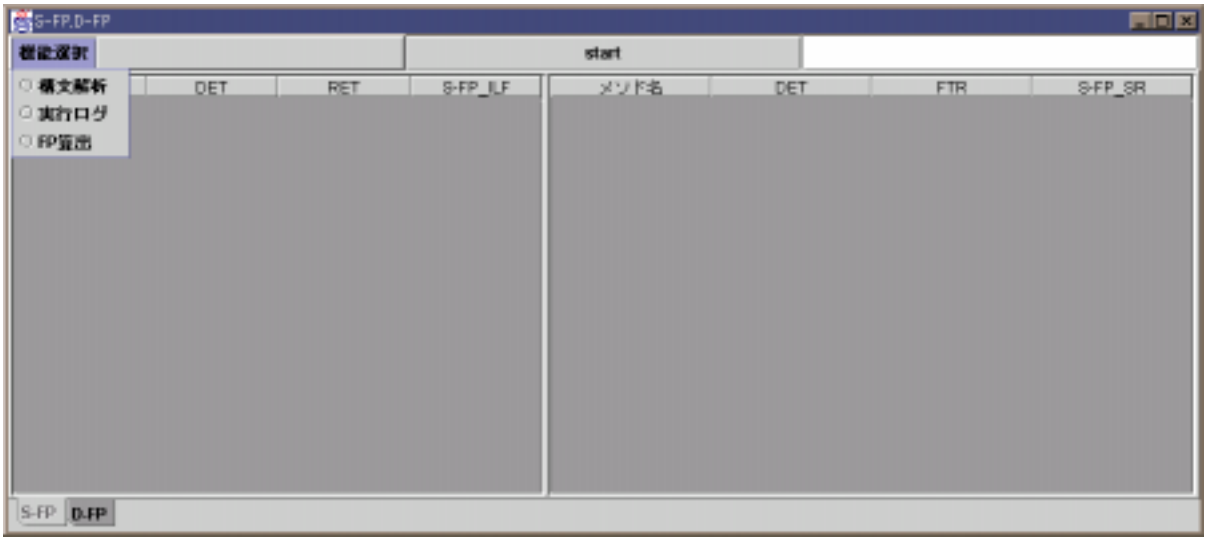

図 7: 機能選択画面

イルを行なう際に標準入力に入力するコマンドと同様の形式でよい.

- 実行コマンド
  実行部を利用する際に計測対象となるプログラムの実行を行なう. 実行部は Java Virtual Machine Profiler Interface(JVMPI) を用いて作成しているので, 実行ログファイルを出力するためには java コマンドに「-Xrunprofiler」というオプションを付け足す必要がある.

- 実行回数
  実行部を利用する際に計測対象となるプログラムの実行を何回連続して行なうかを設定を行なう. 特に指定がなければ 1 回だけ実行する.

の設定を行なうことができるが, 構文解析機能の使用では,「Java プログラムが存在するフォルダ」「ログファイルの出力フォルダ」の二つの指定を行なえばよい.

4.5.2 実行部

実行部は, [Input compile & execute command] ダイアログにおいて,「Java プログラムが存在するフォルダ」「ログファイルの出力フォルダ」の二つの指定を行なう. 計測対象システムを複数回実行させてログファイルを得る場合には, ツール上で「コンパイルコマンド」「実行コマンド」「実行回数」の指定も行なう. しかし, 実行回数が 1 回の場合はこれらの指定は省略して標準入力上でコンパイル, 実行を行なえばよい. 実行を行なう際には java コ

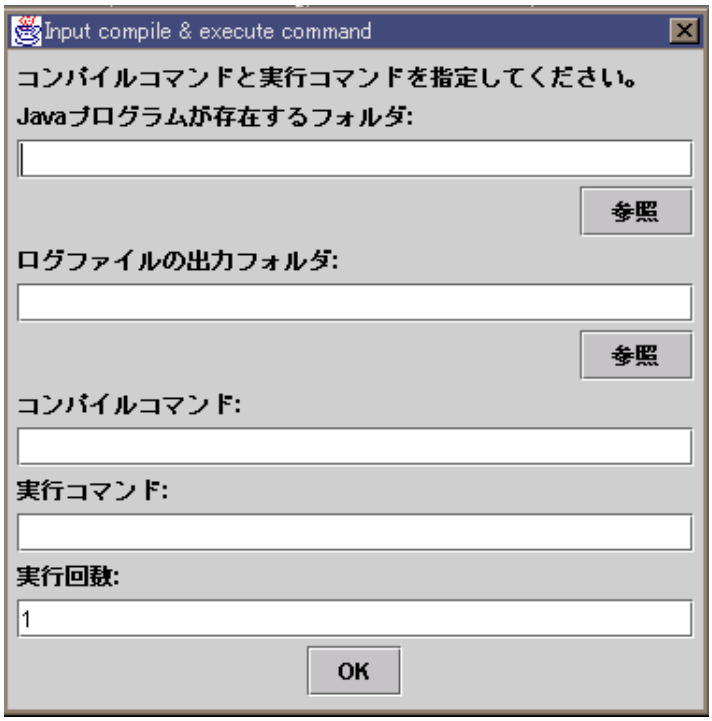

図 8: 構文解析と実行機能の入力画面

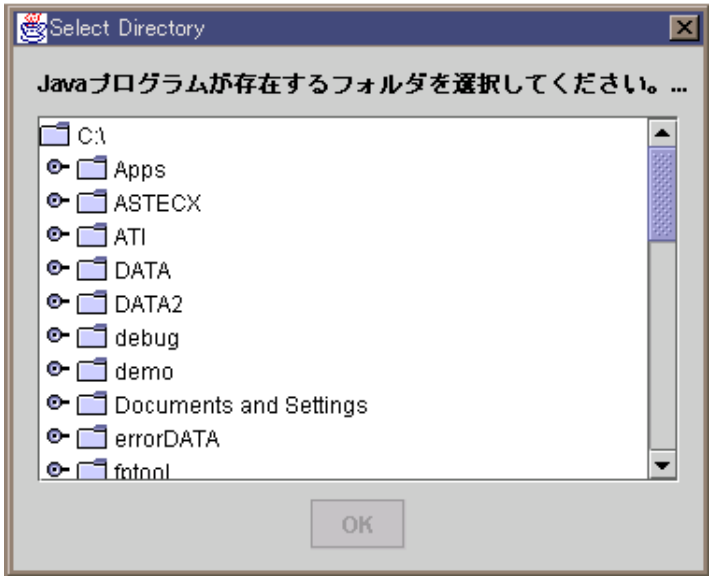

図 9: フォルダ選択画面

マンドに「-Xrunprofiler」オプションを利用する．

4.5.3 FP 計測部

機能選択から FP 算出を選択する．そして「start」ボタンを押すと [analysis file & log file] ダイアログ (図 10) が表示される．このダイアログでは，FP 計測対象システムについて，構文解析部の出力である「構文情報ファイル」と実行部の出力である「実行ログファイル」の指定を行なう．参照ボタンを押すと，[Select File] ダイアログ (図 11) が表示されるので，これを利用してファイル名を入力することができる．

構文情報ファイルと実行ログファイルの指定を行なうと，図 12 に示すダイアログが表示される．ここでは計測対象システムを構成する全クラスについて，構文解析情報ファイルからクラス名とパッケージ名を取得し，それらのクラスからユーザが DF クラスの指定を行なう．

DF の指定を行なうと，図 13 のように FP 計測結果に関する情報が表示される．ここでは以下の情報を表示する．画面左側について，

- クラス名：データファンクションに指定されたクラス名をパス付きで表示.
- 種類：データファンクションの種類 (内部論理ファイルまたは外部インターフェースファイル).
- DET：データ項目数.
- RET：レコード種類数.
- D-FP_DF：そのデータファンクションの FP 値.

画面右側について，

- トランザクション名：トランザクションファンクション名は，トランザクションファンクションとなる一連のメソド呼び出しの最初のメソド呼び出しとなる境界クラスのメソド呼び出しを利用して表す．表示形式は,「クラス名. メソド名. 引数の個数」である.
- 種類：トランザクションファンクションの種類 (外部入力，外部出力，外部照会).
- DET：データ項目数
- FTR：関連ファイル数
- D-FP_TF：そのトランザクションファンクションの FP 値

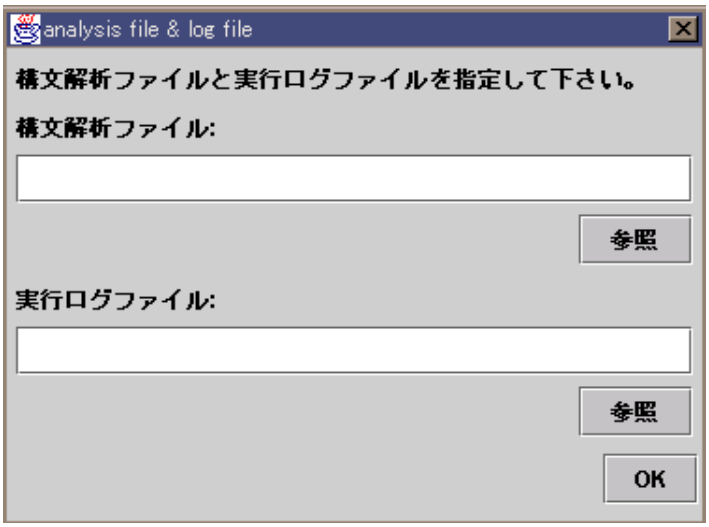

図 10: FP 計測機能の入力画面

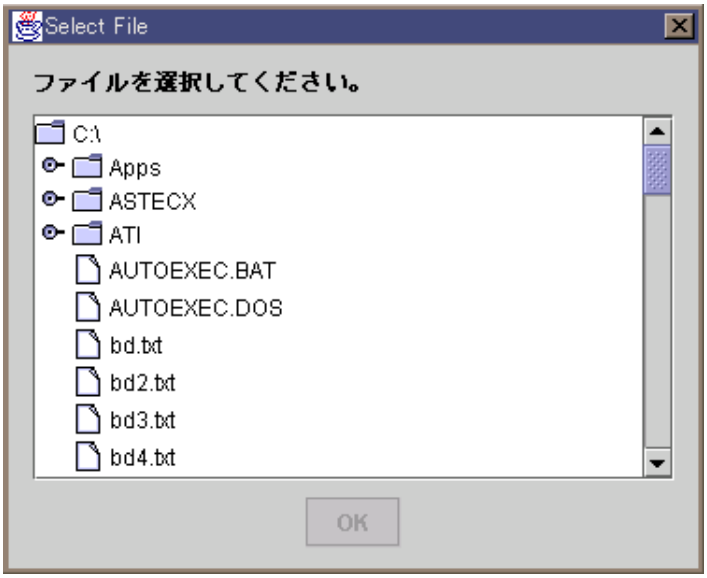

図 11: ファイル選択画面

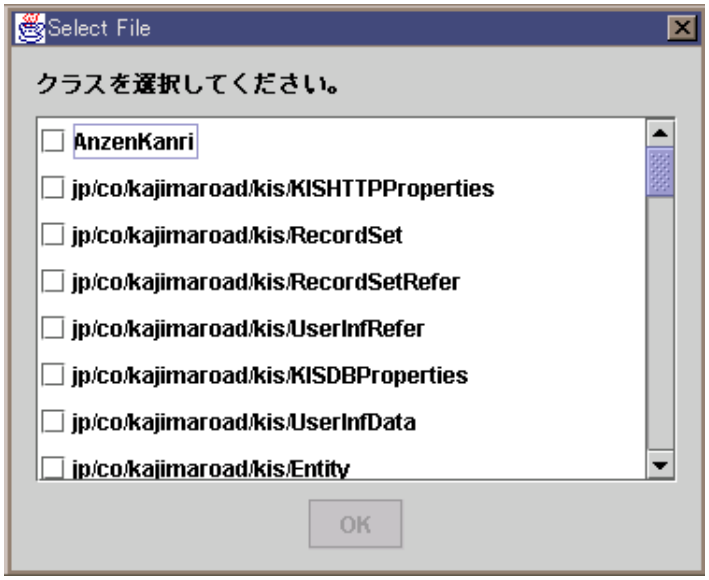

図 12: データファンクション選択画面

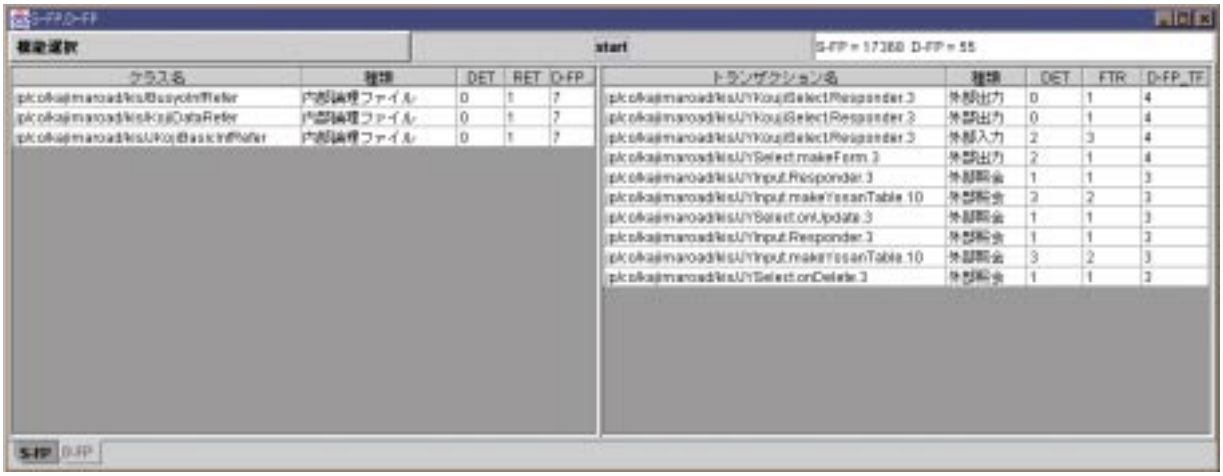

図 13: 計測結果画面

5　評価実験

5.1　実験の概略

提案した手法の有効性を評価するために，あるシステムに対してツールで計測したファンクションポイント値と開発現場のファンクションポイント計測の熟練者がそのシステムの要求仕様書から計測したファンクションポイント値との比較を行なった．

評価実験の対象システムには，ある企業で開発された Web アプリケーションの基幹情報システムを用いた．このシステムは標準的なオブジェクト指向の手法を用いて開発されており，今後似たような Web アプリケーションの開発が多く行なわれると考えられるため，評価対象として妥当であると判断した．システムの規模は約 10K ステップである．その構成を図 14 に示す．

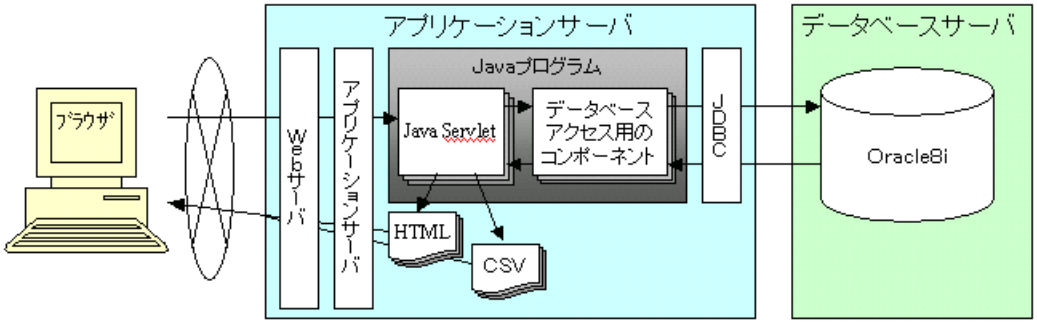

図 14: 評価対象システムの構成

5.2　実験準備

提案手法を用いてファンクションポイントの計測を行なう際，ユーザが指定しなければならないものとして，データファンクションクラスと境界クラスがある．これらの指定は以下の方針で行なった．

- データファンクションクラスの指定
  計測対象のシステムでは，データベースのテーブルへのアクセスをカプセル化している．そこで，これらのカプセル化されたクラスをデータファンクションとして指定した．

- 境界クラスの指定
  境界クラスの選び方として以下の 3 通りの選び方を用い，それぞれの選び方によるファンクションポイント値と要求仕様書からの計測値の比較を行なうことにした．

(条件 A) Java サーブレットクラスのすべてを境界クラスとした.

(条件 B) 他のサーブレットクラスから呼び出される取得データ表示用のサーブレットや確認メッセージ表示用のサーブレット以外を境界クラスとした.

(条件 C) (条件 B) で指定したサーブレットから, 他のサーブレットから呼び出されるデータ入力用のサーブレット以外を境界クラスとした.

次節以降で, 以上で述べた 3 通りの境界クラスの指定方法で行なった計測結果と要求仕様書から熟練者が計測した計測結果の比較を行なう.

5.3 実験結果

5.3.1 データファンクションの計測結果

データファンクションとなるクラスの指定については, データベースへアクセスし, データを受け渡しするメソドを実装したクラスを指定する必要がある. 計測対象のシステムではデータベースへアクセスする処理を隠蔽しているクラスが存在するので, それらのクラスをデータファンクションとして指定した. ツールによるデータファンクションの計測結果とファンクションポイント計測の熟練者が仕様書から計測した結果との比較を表 7 に示す.

表 7: データファンクション計測結果

		ソースコードからの計測	要求仕様書からの計測
DF 数	EIF	1	7
	ILF	11	4
	合計	12	11
FP 値合計		75	65

EIF: 外部インターフェースファイル ILF: 内部論理ファイル

5.3.2 トランザクションファンクションの計測結果

トランザクションファンクションについては, 5.2 で述べた (条件 A), (条件 B), (条件 C) の 3 種類の条件の下で計測を行なった. このときデータファンクションに指定したクラスは 5.3.1 の計測時に指定したものと同じで, これは 3 つの条件すべてに共通である. ツールによるトランザクションファンクションの計測結果と熟練者が仕様書から計測した結果との比較を表 8 に示す.

表 8: トランザクションファンクション計測結果

		(条件 A)	(条件 B)	(条件 C)	要求仕様書からの計測
TF 数	EQ	168	1	0	0
	EO	21	23	13	14
	EI	14	9	9	6
	合計	203 (19)	33 (19)	22 (19)	20
FP 値合計		660 (79)	148 (79)	99 (87)	105

EQ: 外部照会　EO: 外部出力　EI: 外部入力

注意) 表中の () 内は抽出した TF のうち仕様書から抽出した TF と対応するものの数

5.3.3 全体結果

ファンクションポイント値全体での比較を表 9 に示す.

表 9: システム全体の計測結果

		(条件 A)	(条件 B)	(条件 C)	要求仕様書からの計測
FP 値	DF の FP 数	75	75	75	65
	TF の FP 数	660	148	99	105
	合計	735	223	174	170

5.4 結果分析

- データファンクション

  データベースへのアクセスを処理するクラスの設計方針によっては，熟練者が仕様書から抽出したデータファンクションとシステムを構成するクラスとを常に対応付けることができるとは限らない．しかし，今回の評価実験においては，データベースへのアクセスを隠蔽するクラスをデータファンクションに指定することで，熟練者による仕様書からの計測値に近い値を得ることができた．この結果は，データファンクションの指定の仕方次第で仕様書からの計測結果に近いファンクションポイントを実装コードから得られる可能性があることを示していると考えられる．

  また，データファンクションのファンクションタイプの分類については，ツールと熟練者の計測結果の誤差が大きくなった．熟練者が要求仕様書から計測すると内部論理

ファイルの個数より外部インターフェースファイルの個数の方が多くなったが，ツールはほとんどのデータファンクションを内部論理ファイルとして分類していた．

- トランザクションファンクション
  境界クラスの指定の条件を (条件 C) とすることで，ツールによって抽出されたトランザクションファンクションの数は，熟練者が仕様書から抽出したトランザクションファンクションの数に近くなった．また，それぞれで抽出されたトランザクションファンクションの多くは対応付けを行なうことができた．
  (条件 C) において，ツールのトランザクションファンクションの数が仕様書からの抽出された数より多くなった原因として，別々の画面で使われている同じトランザクションファンクションについて，熟練者が仕様書から抽出する場合は同じトランザクションファンクションとして抽出していたが，ツールでは別のトランザクションファンクションとして抽出していたことが挙げられる．
  また，データを PDF ファイルや CSV ファイルなど，ファイル形式でダウンロードするようなトランザクションファンクションについては，トランザクションファンクションのデータ項目数の値が正確に計測できなかったため，複雑度が低くなりファンクションポイント値が少なく計測された．

  また，(条件 C) において抽出されたトランザクションファンクションのファンクションタイプの分類については，ツールと熟練者の分類結果は近いものとなった．

5.5 考察

ツールの計測結果と熟練者が要求仕様書から計測した計測結果との間に誤差が生じていた．ここでは計測誤差の原因とその対処方法について考察する．

内部論理ファイルと外部インターフェースファイルの分類では，要求仕様書からの計測における熟練者の分類結果に比べて，ツールが多くのデータファンクションを内部論理ファイルとして分類していた．この問題点についての対処方法として，次の 2 つが考えられる．

- ユーザがツール上で直接ファンクションタイプを指定できるようにする．
- 構文解析時にソースコードの意味解析も同時にしておき，データファンクションに指定されたクラスの保持するデータが，システム実行中に実際に書き換えられたかどうかを調べてファンクションタイプを決定することで，ファンクション分類の精度を高める．

また，別々の画面で使われている同じトランザクションファンクションをツールでは別のトランザクションファンクションとして抽出していた．この問題点についての対処方法とし

て，トランザクションファンクションを構成する一連のメソド呼び出しのうち，共通部分がある一定量以上あれば同一トランザクションファンクションとする計測ルールを追加することが考えられる．

6 まとめ

本研究では，ファンクションポイント法の主流技法である IFPUG 法のルールに基づいて，Java ソースコードからファンクションポイント値を計測するための計測ルールを提案し，その計測ルールに基づいてファンクションポイント計測ツールを開発した．また，実際の開発現場で開発されたシステムに対してツールを適用し，ツールの計測値とファンクションポイント計測の熟練者が要求仕様書から計測した値との比較評価を行なった．

今後の課題として，以下のような点がある．

- 熟練者による要求仕様書からの計測値とツールの計測値との誤差の原因となっていた計測ルールの改良を行なう．具体的には，データファンクションが内部論理ファイルであるか外部インターフェースファイルであるかの識別方法や別々の画面で使われている同一トランザクションファンクションを，別のトランザクションファンクションとして計測していた点などがある．

- 本手法の IFPUG 法に関する部分は 1994 年の Function Point Counting Practices Manual(Release 4.0)[6] に基づいて提案しているが，1999 年に Release 4.1[7] が発表され，ファンクションポイント計測に関するいくつかのルールが細かい部分で変更を加えられている．本提案手法がそれに適合しているかどうかを考察し，必要があれば提案手法，計測ツールの改良を行なわなければならない．

- これまでに開発されているファンクションポイント計測ツール [13][18] と統合する．

謝辞

本研究において，常に適切な御指導および御助言を賜わりました大阪大学大学院基礎工学研究科情報数理系専攻 井上 克郎 教授に心より深く感謝致します.

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基礎工学研究科情報数理系専攻 楠本 真二 助教授に深く感謝致します.

本研究において，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基礎工学研究科情報数理系専攻 松下 誠 助手に深く感謝致します.

本研究に関して御協力頂きました株式会社 日立システムアンドサービス，津田 道夫氏，高橋 まゆみ氏，松下 幸嗣氏，森本 修馬氏に深く感謝致します.

最後に，その他様々な御指導，御助言等を頂いた大阪大学大学院基礎工学研究科情報数理系専攻井上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, in Jhon J.Marciniak, editor, Encyclopedia of Software Engineering, Vol.1, John Wiley & Sons, pp.518-524, (1994).

[2] G. Caldiera, G. Antoniol, R.Fiutem and C. Lokan:"Definition and Experimental Evaluation of Function Points for Object-Oriented Systems", Proceedings of 5th International Software Metrics Symposium, pages 167-178(1998).

[3] G. Caldiera, G. Antoniol, R.Fiutem and C. Lokan:"A Function Point-like Measure for Object-Oriented Software", Journal of Empirical Software Engineering, 4(3):263-287(1999).

[4] G. Caldiera, C. Lokan, G. Antoniol, R. Fiutem, S. Curtis, G. La Commare, E. Mambella:"Estimating Size and Effort for Object Oriented Systems", Proceedings of 4th Australian Conference on Software Metrics, pages 145-158(1997).

[5] J. Gosling, B. Joy, and G. Steele: The Java Language Specification, Addison-Wesley(1996).

[6] IFPUG: "Function Point Counting Practices Manual, Release 4.0", International Function Point Users Group(1994).

[7] IFPUG: "Function Point Counting Practices Manual, Release 4.1", International Function Point Users Group(1999).

[8] 今川 勝博: "要求分析支援システム REQUARIO で作成された仕様書からのファンクションポイント計測ツールの改良", 大阪大学基礎工学部情報工学科 特別研究報告 (1999).

[9] 今川, 柏本, 楠本, 井上, 鈴木, 湯浦, 津田: "要求仕様書からのファンクションポイント計測ツールの改良 ー要求分析ツール REQUARIO で作成された要求仕様書を対象としてー", 全国大会講演論文集 (1), 1-301 ー 1-302(1999).

[10] 今川, 楠本, 井上: "Java プログラムからのファンクションポイント計測に関する一考察", 電子情報通信学会技術研究報告, SS2000-42〜52, Vol.100, No.570, pp.25-32(2001).

[11] I. Jacobson, M. Christerson, P. Jonsson and G. Overgaard: Object-Oriented Software Engineering - A Use Case Driven Approach -, Addison-Wesley(1992).

[12] B.A.Kitchenham:"The Problem with Function Points", IEEE Software, Vol.14, No.2, pp.29-31, March/April, (1997).

[13] 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).

[14] 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，January，(1990).

[15] 中村 永:“科学技術計算とリアルタイム制御に向くソフト計測手法”，日経エレクトロニクス，No.658，pp.175-185(1996-03).

[16] C.Symons:“Software Sizing and Estimating”，John Wiley & Sons(1991).

[17] C.Symons:“Function Point Analysis: Difficulties and Inprovements”, IEEE Transactions on Software Engineering, Vol.14, No.1, pp.2-10，January，(1988).

[18] T.Uemura，S.Kusumoto and K.Inoue: “Function Point Measurement Tool for UML Design Specification”, Proceedings of the Sixth International Software Metrics Symposium, pp.62-69, (1999).