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

# プログラム実行履歴を用いたトランザクションファンクション抽出手法

森岡 佑† 谷口 考治† 楠本 真二† 井上 克郎†
英 繁雄‡ 芝元 俊久‡ 前田 憲一‡ 津田 道夫‡

†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
‡株式日立システムアンドサービス 〒108-8250 東京都港区港南 2-18-1

E-mail: †{ymorioka, kou-tngt, kusumoto, inoue}@ist.osaka-u.ac.jp,
‡{s-hanabusa, t-shibamoto, ke-maeda, m-tsuda}@hitachi-system.co.jp

**あらまし** ファンクションポイント(FP)法とは，ソフトウェアの持つ機能数を基に規模を測定する手法である．通常，ファンクションポイント値は仕様書から計測される．しかし，FP の測定では測定者の判断が入るため，同一の仕様書からの計測であっても誤差が出ることが指摘されている．また，FP を工数見積等に利用するためには，過去に開発されたソフトウェアからの FP 計測を行う必要があり，最終的に実装されたプログラムからファンクションポイント値を計測する手法が求められている．ファンクションポイント値を計測するには，そのソフトウェア中で利用されるデータを表すデータファンクションと，データファンクションへの入力またはデータファンクションからの出力処理を表すトランザクションファンクションの特定が必要である．そこで本研究では，データファンクションと必要なテストデータが与えられるという前提で，プログラムの実行履歴とデータ依存関係解析結果を組み合わせることによってトランザクションファンクションを特定する方法を提案する．更に実プログラムに提案手法を適用し，有用性の確認を行う．

**キーワード** ファンクションポイント，トランザクションファンクション，プログラム実行履歴

# Extraction of Transactional Function Using Method Execution Trace Information

Yu MORIOKA† Koji TANIGUCHI† Shinji KUSUMOTO† Katsuro INOUE†
Shigeo HANABUSA‡ Toshihisa SHIBAMOTO‡ Kenichi MAEDA‡ and Michio TSUDA‡

†Graduate School of Information Science and Technology, Osaka University 1-3, Machikaneyama-cho, Toyonaka, Osaka, 560-8531, Japan
‡Hitachi Systems & Services, Ltd. 2-18-1 Konan, Minatoku, Tokyo, 108-8250 Japan

E-mail: †{ymorioka, kou-tngt, kusumoto, inoue}@ist.osaka-u.ac.jp,
‡{s-hanabusa, t-shibamoto, ke-maeda, m-tsuda}@hitachi-system.co.jp

**Abstract** Function-point (FP) is a measure of software system size based on the functions of the system. Usually, it is calculated from the design documents. However, it has been reported that since FP measurement involves judgment on the part of the measurer, differences for the same product may occur even in the same organization. Also, if an organization tries to introduce FPA, FP will have to be measured from the past software developed there, and this measurement is cost-consuming. To calculate FP, we need to identify data function which shows the data used in the software, and transaction function which shows the data input and output activities to/from data function. This paper presents a method for identifying transaction function using method execution trace information of a program and data dependency analysis, provided the data function and required test data is given. The proposed method is applied to actual software system and its effectiveness is confirmed.

**Keyword** Function-Point, Transaction Function, Execution Trace

## 1. はじめに

ソフトウェアの機能的な規模を見積もる手段の 1 つとして，Albrecht によってファンクションポイント法が提案された[1]．ファンクションポイントは，ソフトウェアの持つ機能の数をもとに，そのソフトウェアの規模を測定する手法であり，ソフトウェアの開発費用

や工数等の見積もりの基礎として利用される.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の関係式を導出する必要がある．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になる．したがって，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値を計測しなければならない.しかし，過去に開発されたソフトウェアには設計仕様書が存在しなかったり，最終成果物で実装されている機能が設計仕様書に反映されていなかったりすることがあり，このような場合にはファンクションポイント値の測定が困難になる.

そこで，本研究では，最終成果物であるプログラム自身からファンクションポイントを測定する手法の開発を目的としている．本稿では，その第一歩として，プログラムの全ての機能を実行するテストデータの集合とデータファンクションの情報を与えられるという条件の基で，プログラムからのトランザクションファンクションの計測手法について検討する.

以降，2. 節ではファンクションポイント法，およびIFPUG 法について説明する．3. 節では Java 言語を用いて開発されたシステムの実行履歴からトランザクションファンクションを計測する手法について，4. 節では提案手法を実際に既存のソフトウェアシステムに適用した実験結果について説明する．5. 節で適用実験の考察，6. 節で関連研究を挙げ，最後に 7. 節でまとめを行う.

## 2. ファンクションポイント

### 2.1. 概要

ファンクションポイント法は，ソフトウェアに含まれている機能規模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．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って加算した値を機能量とする.

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われ，複数の計測手法が存在する[2][3][4]．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 が定めている IFPUG 法によるファンクションポイントの計測を対象として，プログラム実行履歴を用いたトランザクションファンクション計測手法について提案をする.

### 2.2. IFPUG 法

IFPUG 法[1]は，Albrecht が提案した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ったファンクションポイント測定手法である. 図 1 で示すように, Step1～Step7 までの処理を経て計測される．これらの Step のうち，ソフトウェアの機能を抽出するのは Step3 のデータファンクション計測と Step4 のトランザクションファンクションの計測であり，この 2 つの Step が IFPUG 法においてもっとも重要であるといえる.

次に，ファンクションポイント計測における重要なステップとなる Step3・Step4，さらに step3,4 の結果を利用する step5 について説明を加え，ファンクションポイント値の具体的な算出法について記述する.

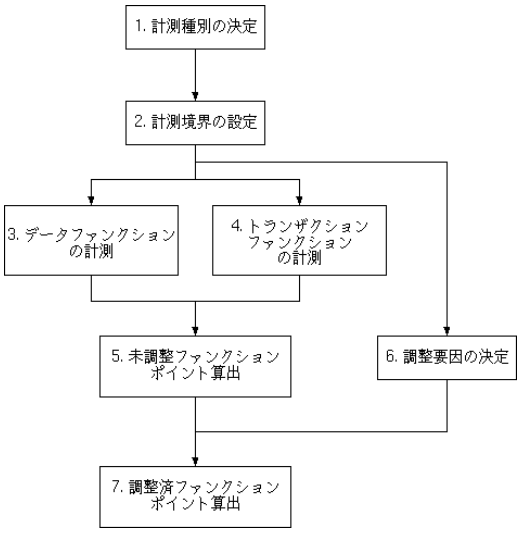


図 1. ファンクションポイント計測手順

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」を指す．Step3 では，アプリケーションの中からデータファンクションを抽出し，ファンクションタイプを与え，複雑さを測定する．データファンクションのファンクションタイプは，計測対象のアプリケーション内で更新される内部論理ファイル(ILF: Internal Logical File) と，計測対象のアプリケーションから更新は行われず参照のみが行われる外部インターフェースファイル(EIF: External Interface File) の 2 種類に分けられる．1 つのアプリケーションには複数のデータファンクションが存在し，それぞれのデータファンクションはレコード種類(RET: Record Element Type)，データ項目(DET: Data Element Type)の 2 つのパラメータによって，低・中・高の 3 段階に重み付けされ，この重みがファンクションの複雑さになる．RET

は ILF や EIF のデータのサブグループ，DET はユーザが識別できるデータ項目を表し，複雑さを決定するために表 1 のマトリックスが使用される．

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 |

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」を指す．したがって，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データ項目に対して行われる処理と捉えることもできる．Step4 では，アプリケーションの中からトランザクションファンクションを抽出し，ファンクションタイプを与え，複雑さを測定する．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は外部入力(EI: External Input)，外部出力(EO: External Output)，外部照会(EQ: External Inquiry)の 3 種類に分けられ，EI はデータファンクションのデータの更新などの入力処理，EO はデータファンクションのデータを加工してユーザに提供する出力処理，EQ はデータファンクションの単純検索情報をユーザに提供する処理と定義されている．出力処理において，算術式・計算・導出データ生成・ILF の維持管理・システムの動作変更の機能を，1 つ以上含む処理を EO，どの機能も含まない処理を EQ とする．データファンクションと同様に，一般的に 1 つのアプリケーションには複数のトランザクションファンクションが存在する．それぞれのトランザクションファンクションは関連するデータファンクション数(FTR: File Type Referenced)と，利用するデータ項目(DET: Data Element Type)の 2 つのパラメータによって，低・中・高の 3 段階に重み付けされる．FTR は，トランザクションファンクションによって読み込みまたは更新が行われる ILF と，読み込みが行われる EIF を表す．トランザクションファンクションの複雑さは，表 2 のマトリックスを用いて決定される．

Step5 未調整ファンクションポイント算出

Step3・step4 で導出した各ファンクションに対して表 3 にしたがって FP 値を計測し，それらを合計して未調整ファンクションポイントを算出する．

表 2. トランザクションファンクションの複雑さ

《 (左)外部入力　(右)外部出力,外部照会 》

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

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

表 3. 未調整ファンクションポイント算出表

《 (左)データファンクション

(右) トランザクションファンクション 》

| type \ 複雑さ | 低 | 中 | 高 |
|---|---|---|---|
| 内部論理ファイル | 7 | 10 | 15 |
| 外部インタフェースファイル | 5 | 7 | 10 |

| type \ 複雑さ | 低 | 中 | 高 |
|---|---|---|---|
| 外部入力 | 3 | 4 | 6 |
| 外部出力 | 4 | 5 | 7 |
| 外部照会 | 3 | 4 | 6 |

## 3. プログラムからのファンクションポイント計測

### 3.1. Java プログラム中のデータファンクション･トランザクションファンクション

プログラムからトランザクションファンクションを抽出するためには，Java プログラム中でデータファンクション，トランザクションファンクションがそれぞれどのように実装されているのかを認識する必要がある．

実際に運用されているシステムを解析すると，データファンクションはしばしば各データファンクションに対応するクラスとして実装される[6]．そして，各データ項目はそのクラスのフィールドとして保持される．一方，トランザクションファンクションはメソッド呼び出し群で実装される[6]．これは，トランザクションファンクションが「データファンクションに対して行われる入出力処理」として捉えられるため，データファンクションのデータを利用して何らかの処理を行うメソッド群をトランザクションファンクションと判断することができるからである．

以上の実装結果を踏まえて，プログラムから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を計測するには，次の 3 段階の処理が必要になると考えられる．

(1) トランザクションファンクションを構成していると考えられるメソッド群を抽出

(2) 各メソッド群によって行われる処理から EI・EO・EQ のうち当てはまる種類に分類

(3) 各メソッド群をトランザクションファンクションと考え，ファンクションポイント算出

本稿は，(1)のトランザクションファンクションを構成していると考えられるメソッド群の抽出を目的とし，その手法について提案するものである．プログラムの実行履歴とデータ依存関係解析結果を組み合わせたメソッド群抽出手法について，次の 3.2.節で説明する．

### 3.2. 提案手法

本研究では，データファンクションと必要なテストデータが与えられるという前提で，プログラムの実行時情報(以下，実行履歴)とデータ依存関係解析結果を組み合わせることによってトランザクションファンクションを特定する方法を提案する．具体的には，データ依存関係解析結果を基にデータファンクションに相当するクラスからのデータフローを実行履歴上にマッピングし，そのデータフロー情報をもとに実行履歴上のメソッド呼び出し群の中からトランザクションファンクションに該当する処理を抽出する．この処理を図示したものが図 2 である．この図は取得した実行履歴をシーケンス図の形式を用いて表したものであり，実線矢印はメソッド呼び出し，破線矢印はメソッドの返り値を表している．なお，本手法はデータファンクションやテストデータを事前に与える必要があるため，適用対象のソフトウェアに対してある程度の知識をもつ者が利用することを前提にしている．

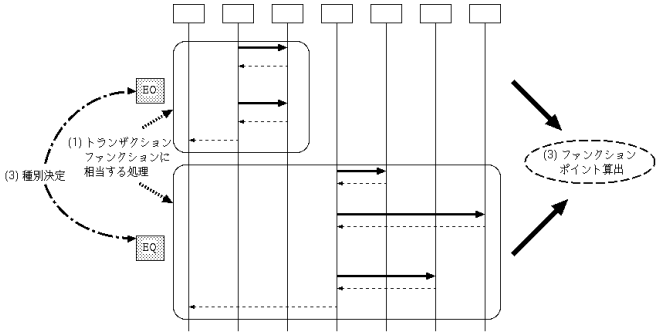


図 2. 実行履歴を用いたトランザクションファンクション計測手順

提案手法の処理手順の流れは以下の通りである．

(1) 計測対象ソフトウェアのトランザクションファンクションを実行し，メソッド呼び出し単位の実行履歴を取得
(2) 静的解析によってデータ依存関係を解析し，入力として与えられたデータファンクションに相当するクラス(以下，DFC)のフィールド値に関連するデータフローを実行履歴上にマッピングする．
(3) 範囲が重複するデータフローをマージ
(4) 得られたデータフローをトランザクションファンクションとして抽出

手順(2)で扱う DFC のフィールド値に関連するデータフローには，DFC のフィールド値が取得されてから利用されるまでのデータフローと，データが入力されてから DFC のフィールド値にセットされるまでのデータフローの 2 種類がある．前者をフォワードフロー，後者をバックワードフローと呼ぶ．本手法では，両方のデータフローについて解析を行う．

手順(2)で得られたデータフローにおいて，DFC のフィールド値に関連するデータフローの中でデータフロー範囲が重複する箇所は，複数の DFC のデータが同時に利用されていることになる．これは，何らかの処理の 1 回分の実行において複数のデータが同時に利用されていると考えられる．そこで本手法では，手順(3)でこのような範囲が重複するデータフロー群をマージし，複数のデータに関連するデータフローを 1 つのデータフローにまとめる．図 3 はデータフロー範囲をシーケンス図上に示したものである．曲線は DFC のフィールド値のデータフロー範囲を表しており，図中のデータフロー2 においてデータフローがマージされている．マージ処理後の各データフロー範囲内に含まれる複数のメソッド群は 1 つ以上の DFC のデータを利用して何らかの処理を行っていると考えられる．そこで，そのようなメソッド呼び出し群を，トランザクションファンクションの 1 回の実行とみなして抽出する．

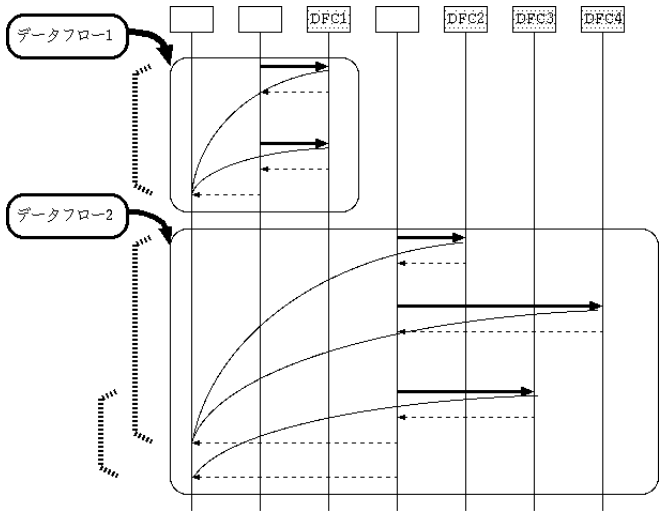


図 3. 重複部分マージ後のデータフロー範囲

## 4. 適用実験

### 4.1. 対象ソフトウェア

実際に運用されている，ソフトウェア開発ツールの貸出管理を行うシステムに対して提案手法を適用した．以下，このソフトウェアをツール貸出システムと呼ぶ．ツール貸出システムには 37 種類のトランザクションファンクションが含まれている．このうち 16 種が一般

ユーザによる操作で実行され,残りの 21 種が管理者のみの操作によって実行される.

今回の実験では，一般ユーザの操作によって実行される 16 種類のトランザクションファンクションに対して提案手法を適用した.

### 4.2. 実験方法

適用実験は以下のような手順で行った．ツール貸出システムにおいて，一般ユーザが実行できる処理を全て行った場合の実行履歴を取得する．取得した実行履歴に対して提案手法を適用し，データフロー範囲を出力する．このデータフロー範囲が設計仕様書に記述されたトランザクションファンクション実行部分と一致するかどうかを確認する.

実行履歴中のメソッド呼び出しには，呼び出された順に初期値を 1 とした ID 番号がつけられており，データフロー範囲はこの ID 番号の範囲で出力される.

### 4.3. 結果の評価方法

実行履歴取得時に設計仕様書とソースコードをもとに，実行履歴中で各トランザクションファンクションが実行されている部分を，メソッド呼び出し ID 番号の範囲として取得しておく．この ID 番号の範囲と提案手法により出力されたデータフロー範囲が 1 対 1 で対応する場合に，トランザクションファンクションが正しく抽出されたと判定する．複数回出現するトランザクションファンクションは，その全ての出現箇所においてデータフローが検出された場合のみ，正しく抽出できたと判定する.

### 4.4. 結果

提案手法によって検出したデータフロー範囲は 32 個であった.このうち 28 個のデータフロー範囲において，実行履歴に現れるトランザクションファンクションと 1 対 1 で対応していた.

次に各トランザクションファンクションに対して,データフロー範囲との関係を示す.含まれる 16 種類のトランザクションファンクションのうち,11 種を正確に抽出することができた．提案手法によって得られたデータフロー範囲と，実際に実行履歴に含まれているトランザクションファンクションを比較し，抽出できたかどうかを示したものが表 4 である.

トランザクションファンクション「物品詳細」・「物品管理品詳細」は，その次に実行されるトランザクションファンクションにもデータファンクションのデータがフローしていた．そのため，それらのトランザクションファンクションに対応するデータフローは，その次に続くトランザクションファンクションに対応するデータフローにマージされてしまっていた．したがって，トランザクションファンクションとデータフロー範囲が 1 対 1 の対応をしていなかった.

トランザクションファンクション「アイテムグループ検索(大分類)」・「アイテムグループ検索(中分類)」・「コードテーブル検索」を表すデータフローは「物品新規登録」に対応するデータフロー範囲に包まれていたため，検出できなかった．これは，複数のトランザクションファンクションが同時に実行されていることが要因となった結果であると考えられる.

表 4. トランザクションファンクション抽出結果

| トランザクションファンクション名 | 判定 |
|---|---|
| 利用中物品検索 | ○ |
| 使用中物品検索(条件付) | ○ |
| 物品検索 | ○ |
| 物品詳細 | × |
| 物品貸出 | ○ |
| 物品登録未承認案件一覧 | ○ |
| 物品新規登録(一般向け) | ○ |
| 物品登録未承認案件詳細 | ○ |
| 物品管理品一覧 | ○ |
| 物品管理品詳細 | × |
| 物品追加登録 | ○ |
| 備考表示 | ○ |
| 備考更新 | ○ |
| アイテムグループ検索(大分類) | × |
| アイテムグループ検索(中分類) | × |
| コードテーブル検索 | × |

## 5. 考察

提案手法では,「データファンクション-クラス」という対応を仮定しているため，得られる結果が対象プログラムの実装に大きく依存することを考慮する必要がある．したがって，ある程度のコード生成規約に基づいて対象プログラムを生成することが求められる．具体的には,(1)プログラム中で扱うデータ項目は対応するクラスのフィールド値にする,(2)データ項目として位置づけられるフィールド値へのアクセスには,getter, setter の利用を推奨する,といった規約に基づくコーディングを求めることになる.このような規約は,複数人による開発における実装方法の統一やコードのモジュール化を進める上でも重要である.

対象プログラムの実装に依存した箇所として他に挙げられるのは,トランザクションファンクション「物品詳細」と「物品管理品詳細」が抽出できなかった点である.対象プログラムのソースコードを解析すると,詳細部で取得した DFC のフィールド値をそのまま次トランザクションファンクションで利用するような実装が行われていたために，対象のトランザクションファンクションを抽出できなかったということが判明した．このような誤検出を防ぐには，各トランザクション実行時に，利用するデータファンクションのデータを取得するような実装規約が必要である.

また，複数のトランザクションファンクションが同

時に実行されている場合，現在のアルゴリズムでは対応できない．解決方法として，データファンクション指定時に，同一トランザクションファンクション中で利用されるデータ項目かどうかを付加情報として与えるという方法が考えられる．重複するデータフローにおいて，同一トランザクションファンクション中で利用されないデータ項目のフローが検出されていた場合には別トランザクションの実行とみなし，データフロー範囲をマージせずに出力すれば同時に進行しているトランザクションファンクションを識別できる．この問題点の解決には，今後も検討および試行が必要である．

## 6. 関連研究

Sneed[7]は，Java 言語以外のプログラミング言語で実装されているソフトウェアに対して，ソースコードからファンクションポイントを計測する方法を提案している．COBOL のような旧来のプログラミング言語に関しては，データの入力・出力，およびデータベースの検知が単純なためファンクションポイント計測が行いやすい．一方，オブジェクト指向言語は旧来の言語に比べて入出力やデータベースの認識が困難であるため，これに伴って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の同定も困難になるという見解を示している．

Antoniol ら[6]は，オブジェクト指向言語で記述されたソフトウェアに対して，独自の “Object-Oriented Function Points (OOFP)” という指標を策定し，機能規模の測定を行っている．「論理ファイル - クラス」,「トランザクション - メソッド」をそれぞれ対応させ，さらにメソッドに関しては入出力の種類を考慮せず，全て “Service Requests (SRs)” として処理する．データファンクションにおけるクラスの集約や継承にも対応しており，構成するクラスの組み合わせによって論理ファイルを全部で 4 種類に分類し，それぞれの組み合わせについてファンクションポイント値を算出している．また，提案手法の全自動化ツールも作成しており，これによってファンクションポイント計測時に問題となる，曖昧さや測定者の主観性を排除し，同一条件における客観的な計測を可能にしている．トランザクションとメソッドを対応させる点が本研究と類似しているが，Antoniol らは 1 個のメソッドを 1 個のトランザクションに対応させているのに対し，我々の提案手法は複数のメソッドを 1 個のトランザクションに対応させている．このことから，提案手法はソースコード中のトランザクションファンクションの実態をより正確に反映していると考えられる．

## 7. まとめ

データファンクションと必要なテストデータが与えられるという前提で，プログラム実行履歴を利用してトランザクションファンクションを特定する手法を提案した．データファンクションのデータに対して処理を行うメソッド群を 1 つのトランザクションファンクションとすることでトランザクションファンクションの抽出を行い，適用実験において手法の妥当性を示した．

今後の課題は，重複したデータフローが検出された場合に，同一トランザクションか，別トランザクションかの判定をできるようにすることである．さらに，他のソフトウェアに対しても適用実験を行い，手法の汎用性を調査することも課題である．最終的には，トランザクションファンクション抽出結果を基にファンクションポイントを算出し，実際に設計仕様書から測定した値との比較も実施したい．

## 8. 謝辞

本研究は一部科学研究費補助金基盤研究 (C)(17500022) の補助を受けている．

## 文　献

[1] IFPUG: “Function Point Counting Practices Manual, Release 4.2”, International Function Point Users Group, 2005.

[2] C.Symons, “Software Sizing and Estimating”, John Wiley & Sons, 1991.

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

[4] G.Antoniol, R.Fiutem, C.Lokan, “Object-Oriented Function Points: An Empirical Validation”, Empirical Software Engineering, vol.8, No.3, pp.225-254, September, 2003.

[5] 谷口考治, 石尾隆, 神谷年洋, 楠本真二, 井上克郎, “Java プログラムの実行履歴に基づくシーケンス図の作成”, ソフトウェア工学の基礎ワークショップ(FOSE2004), pp.5-15, November, 2004.

[6] G.Antoniol, C.Lokan, G.Caldiera, R.Fiutem, “A Function Point-Like Measure for Object-Oriented Software”, Empirical Software Engineering, vol.4, No.3, pp.263-287, September, 1999.

[7] Harry M.Sneed, “Extraction of Function-Points from Source-Code”, proceedings of the 10th International Workshop on New Approaches in Software Measurement, pp.135-146, October, 2000.