修士学位論文

題目

ファンクションポイント計測のための実行履歴からトランザクションファンクションを抽出する手法の提案

指導教員
井上 克郎 教授

報告者
森岡　佑

平成 19 年 2 月 13 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 ソフトウェア工学講座

平成 18 年度 修士学位論文

ファンクションポイント計測のための実行履歴からトランザクションファンクションを抽出する手法の提案

森岡　佑

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 (FP) 法が用いられることが増えてきている．ファンクションポイント法とは，ソフトウェアの持つ機能数を基に，定量的にソフトウェアの規模を測定する手法である．ファンクションポイントを工数見積等に利用するためには，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導出する必要がある．通常，ファンクションポイント値は設計仕様書から計測されるが，過去のソフトウェア開発においては，設計仕様書が既に存在せず，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く，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．そして，実際に運用されているプログラムに提案手法を適用してトランザクションファンクションを抽出し，設計仕様書から得られたトランザクションファンクションとの比較を行い，提案手法の有用性を確認した．

主な用語

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

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 [1]．通常，開発工数や開発期間を予測するのに，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 [2] が提案された．現在，これをベースに IFPUG 法 [3] が考案され，広く実用されている．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．ファンクションポイントを測定するには，ソフトウェアの機能として，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特定が必要になる．データファンクションは対象のソフトウェア中で利用されるデータを表し，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．

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ため，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い場合や，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い場合があり，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となる．したがって，最終成果物であるプログラムからファンクションポイントを測定する方法が求められている．

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，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ケースの実行履歴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もとで，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，プログラムからファンクションポイントを測定する方法について説明を行う．

4 節では，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トランザクションファンクション抽出手法を提案し，さらに 5 節で実プログラムに対して提案手法の適用実

験を行って提案手法の評価を行い，考察を加える．6 節で関連研究を挙げ，最後に 7 節で本研究の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

この節ではファンクションポイント法の概要を述べ，ファンクションポイントの標準的な測定法として現在広く普及している IFPUG 法による測定手順と，ファンクションポイントを開発現場に導入する際に発生する課題について説明する．

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，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．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として，(1) 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，(2) 規則に従って計測される値であるため，計測者に依存せず一定の値が得られる，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われ，複数の計測手法が存在する [4][5][6]．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 [3] 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．IFPUG 法によるファンクションポイント値は，図 1 で示すように，step1〜step7 までの処理を経て計測される．

2.2.1　Step1　計測種別の選択

計測種別を以下のの 3 種類から選択する．計測種別によって，Step2 で設定するファンクションポイント計測境界が異なる．

- 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

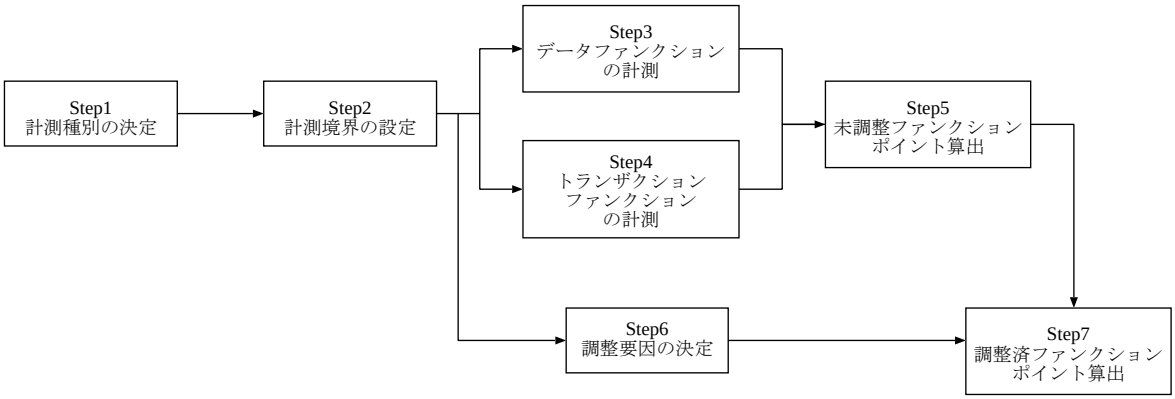

図 1: ファンクションポイント計測手順

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.

2.2.2 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，1 つの境界はユーザから見て機能面で閉じていなければならない.

2.2.3 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には以下の 2 種類のファンクションタイプがある.

- 内部論理ファイル (ILF: Internal Logical File)
  計測対象のアプリケーション内で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 (EIF: External Interface File)
  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データファンクション.

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数，データ項目数の 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．

- データ項目数 (DET: Data Element Type)
  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 (RET: Record Element Type)
  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．内部論理ファイルや外部インターフェースファイルのデータのサブグループで構成される．

データファンクションに付けられた重みがファンクションの複雑さを表す．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．

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

2.2.4 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は以下の 3 種類のファンクションタイプに分けられる．

- 外部入力 (EI: External Input)
  データファンクションのデータの更新などの入力処理．
- 外部出力 (EO: External Output)
  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．

- 外部照会 (EQ: External Inquiry)
  データファンクションのデータを加工せず，単純検索情報で得られるデータをユーザに提供する処理

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 にそれぞれ分類する．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，利用するデータ項目数と，関連するファイル数の 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．

- データ項目数 (DET: Data Element Type)
  入出力されるデータ項目の個数．
- 関連ファイル数 (FTR: File Type Referenced)
  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．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 が対象になる．

トランザクションファンクションにつけられた重みがファンクションの複雑さを表す．複雑さを決定するために，表 2，3 を使用する．

表 2: 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

FTR: 関連ファイル数 DET: データ項目数

2.2.5 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 4，5 をもとにファンクションポイント値を計測し，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 3: 外部出力・外部照会の複雑さ

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

表 4: データファンクションの未調整ファンクションポイント算出表

タイプ \ 複雑さ	低	中	高
ILF	7	10	15
EIF	5	7	10

2.2.6 Step6 調整要因の決定

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 6 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．

2.2.7 Step7 調整済ファンクションポイント算出

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

表 5: トランザクションファンクションの未調整ファンクションポイント算出表

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

表 6: 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

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

Step1〜Step7 のうち，ファンクションポイント計測において最も重要なのは，ソフトウェアの機能を抽出する 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る．

2.3 ファンクションポイント導入時の課題

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おい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，ファンクションポイントの計測には測定者の主観的な判断が必要になる．その結果，同一ソフトウェアに対する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 [7]．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 [8].

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．したがって，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．さらに，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．

そのため，最終成果物であるプログラムのソースコードからファンクションポイントを測定する手法が求められている．本研究では，プログラムからファンクションポイントを測定する手法の第一歩として，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もとで，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手法の提案を行う．3 節では，Java プログラムからファンクションポイントを測定する方法について説明し，さらに 4 節で Java 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，トランザクションファンクション抽出手法を提案する．

3　プログラムからのファンクションポイント計測

3 節では，はじめに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プログラム中での実装方法について述べる．次に，各ファンクションの実装形態を基に，ファンクションポイントを自動的に計測するための方針とその手順について説明する．

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しておく必要がある．

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 [9]．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のまとまりで実装される [9]．これは，トランザクションファンクションを「データファンクションに対して行われる入出力処理」として捉えることができるため，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．

3.2　想定するファンクションポイント計測方針

Antoniol ら [9] によるファンクションポイント計測時の定義と同様に，データファンクションをクラス，トランザクションファンクションをメソッド呼び出しのまとまりにそれぞれ対応させる．トランザクションファンクションに対応させるメソッド呼び出し群は，データファンクションに対して何らかの処理を行うメソッドのまとまりと定義する．

我々が想定するプログラムからファンクションポイントを計測する際の前提として、データファンクションに対応するクラスはユーザが指定する。以降、データファンクションとして利用されるクラスをデータファンクションクラス (DFC) と呼ぶ．トランザクションファンクションを計測するには、プログラムの動的情報を用いる。静的情報を用いる手法も考えられるが、オブジェクト指向プログラムは動的束縛等によって実行経路が実行時に決められることや、トランザクションファンクションの開始点および終端点が静的情報を用いる場合よりも決定しやすい点を考慮し、動的情報を利用する。

3.3 節で、プログラム実行履歴を利用したファンクションポイント計測法の詳細な手順を説明する。

トランザクションファンクションとなるメソッド呼び出し群と、必要なパラメータをプログラムの実行履歴から抽出し，あらかじめユーザが指定したデータファンクションクラスの

情報を組み合わせることによって、ファンクションポイント値を算出することを方針とする.

3.3 プログラムからファンクションポイントを計測する手順

プログラム中で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の実装結果, および計測方針を踏まえると，次の 8 段階の処理を実行することでプログラムからファンクションポイントを計測することができると考えられる.

(1) データファンクションクラスを指定，およびプログラムを実行して履歴を取得

(2) 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呼び出し群を抽出

(3) メソッド呼び出し群によって実行される処理から，各メソッド呼び出し群を EI・EO・EQ に分類

(4) 指定したデータファンクションクラスを基にデータファンクションのファンクションポイントを計測

(5) 各メソッド呼び出し群で使用しているデータファンクションクラスとそのフィールド値の個数を求める

(6) 類似したメソッド呼び出し群をまとめる

(7) 各メソッド呼び出し群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

(8) (4),(7) の測定結果から未調整ファンクションポイントを計測 (図 2)

処理 (2) によって抽出された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候補として捉え，メソッド呼び出し群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値を測定し，あらかじめユーザが指定したデータファンクションクラスから求めたデータファンクションのファンクションポイント値と合計して，未調整ファンクションポイントを計測する．実行履歴中に同一トランザクションファンクションが複数回実行される場合があるが、処理 (6) で類似したメソッド呼び出し群をまとめるため、メソッド呼び出し群抽出の際は重複を考えなくてもよい。

このプログラムからのファンクションポイント計測処理を実行する際にユーザが行うのは、処理 (1) のみであり，処理 (2) 以降は自動化することが可能である．自動化する部分を増やすことにより，ファンクションポイントにおいて問題となる、測定者の主観的な判断を必要とする部分を減らし，客観的なファンクションポイント計測ができる．また，上記の処

理手順で必要とするのは計測対象プログラムのみであり，設計仕様書が存在しないプロダクトに対してもファンクションポイントを計測できる．

本論文における研究は，処理手順 (2) 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呼び出し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呼び出し群抽出手法について，4 節で説明する．

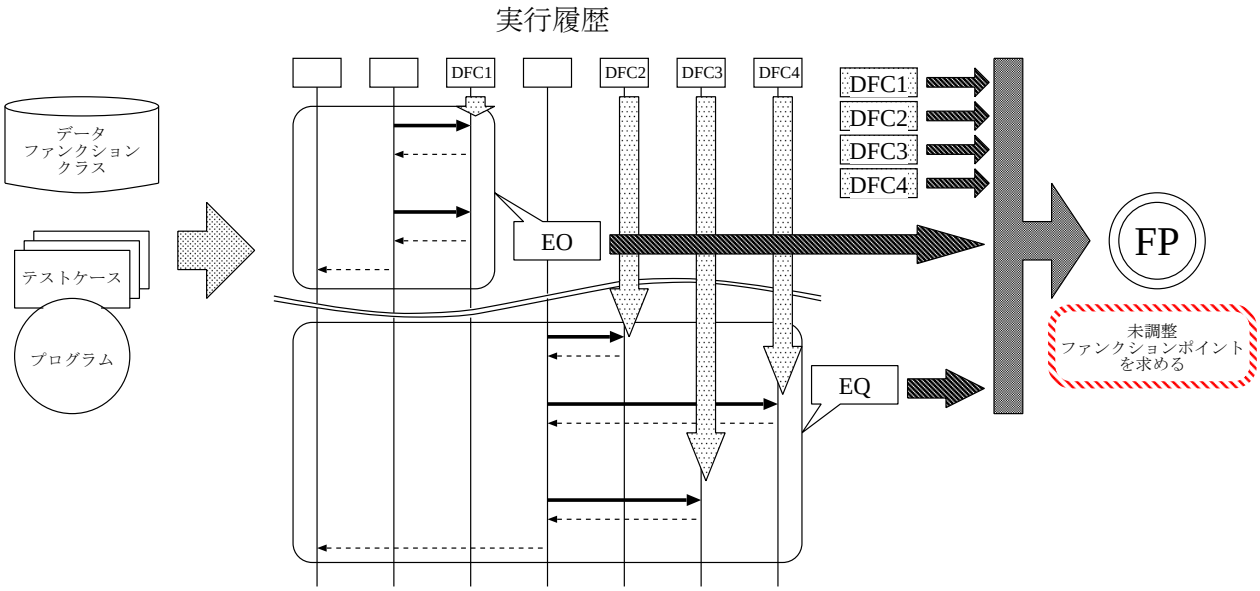

図 2: 未調整ファンクションポイント計測

3.4 プログラムを利用したファンクションポイント計測法の特徴

上記手順でファンクションポイントを計測する際、得られるファンクションポイント計測結果は対象プログラムのソースコードの実装に依存する。ファンクションポイント値を計測するには、ソフトウェアの設計図や要求仕様書、ソースコードといった媒体が利用される。ファンクションポイントの前提として、どの媒体を用いても同じファンクションポイント値が得られるが、本節で我々が提案するファンクションポイント計測法では、データファンクションやトランザクションファンクションが冗長に作成されている場合、重複する部分であっても別のファンクションとして認識する。同一のファンクションを異なる実行系列において 2 重に実装していた場合、このファンクションに対するファンクションポイント値は 2 倍になる。

プログラムからファンクションポイント値を測定する本来の目的は、開発工数や開発期間

の見積もり精度を向上させるために、過去に開発されたソフトウェアのファンクションポイント値を基礎データとして蓄積することであるが、この目的以外にも、本節で説明したファンクションポイント計測法を利用できる。例えば、開発工程の初期段階でのファンクションポイント値と、開発終了後に成果物から測定したファンクションポイント値の差を分析することによって、成果物に冗長な機能が含まれている場合や、実装予定であった機能が実装されていない場合の確認や、ファンクションポイントを測定したい部分に関連のあるデータファンクションクラスのみを指定し、ソフトウェアの部分的なファンクションポイント値測定が挙げられる。

4　トランザクションファンクション抽出手法

この節では，本論文で提案するトランザクションファンクション抽出手法について説明する．4.1 で提案手法の概要を述べ，次に 4.2 で提案手法で行うデータフローを実行履歴上にマッピングする処理について，さらに 4.3 で範囲が重複するデータフローをマージする処理について，それぞれ説明する．

4.1　提案手法概要

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，メソッド呼び出しが記録された実行履歴を取得し，メソッド単位のデータ依存関係を静的解析した後，データファンクションに相当するクラス (DFC) のフィールド値からのデータ依存関係 (データフロー) を決定し，得られた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 3 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．

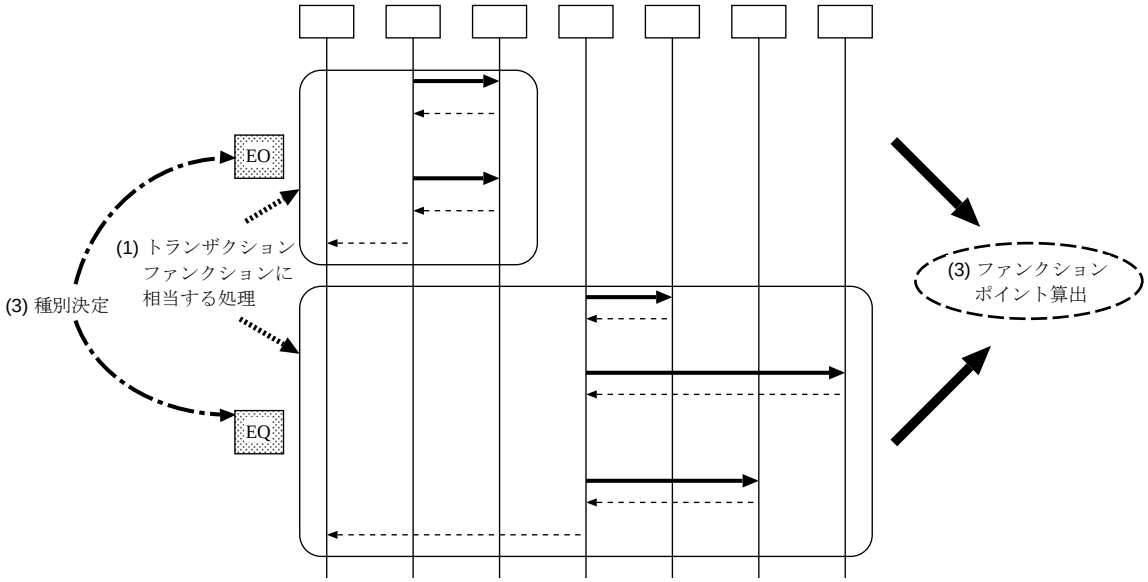

図 3: 実行履歴を用いたトランザクションファンクション抽出手順

実行履歴から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呼び出し群を抽出する手法の詳細な処理手順は以下の通りである．

(1) メソッド呼び出しが記録されている実行履歴を取得後，メソッド単位のデータ依存関係を静的解析し，実行時のデータファンクションクラスからのデータフローを決定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

(2) 範囲が重複するデータフローをマージ

(3) 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

本手法で用いる実行履歴には，実行履歴からシーケンス図を描画するツール Amida[10] を利用する際に得られる実行履歴を使用する．この実行履歴中に，実行時に動作しているオブジェクトと呼び出されているメソッドが記録されている．

以降，4.2 でデータフローのマッピングについて，4.3 ではデータフローのマージについて，それぞれ説明を加える．

4.2 データファンクションクラスからのフィールドに関連するデータフローの検出

データファンクションクラスからのフィールド値に関するデータフローを解析し，実行履歴上でのデータフロー範囲を決定する．データファンクションクラスのフィールド値に関連するデータフローには，データファンクションクラス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データファンクションクラス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い，結果を統合してデータフローを判定する．

図 4 はデータフローのマッピングをシーケンス図上に示したものである．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，曲線はデータファンクションクラスのフィールド値のデータフロー範囲を表している．

4.3 データフローのマージ

実行履歴へのマッピングによって得られたデータフローにおいて，データファンクションクラス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，複数のデータファンクションクラスのデータが同時に利用されていることを表している．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ために起こると考えられる．そこで本手法では，手順 (2) でこのような範囲が重複するデータフロー

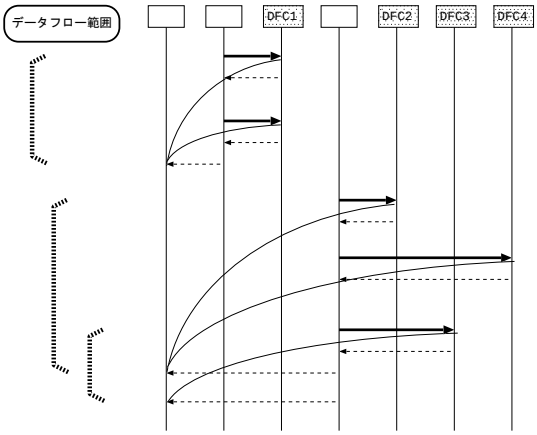

図 4: データフロー範囲

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 5 はマージした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．図 4 と同様に曲線はデータファンクションクラス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 つ以上のデータファンクションクラス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，こ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．

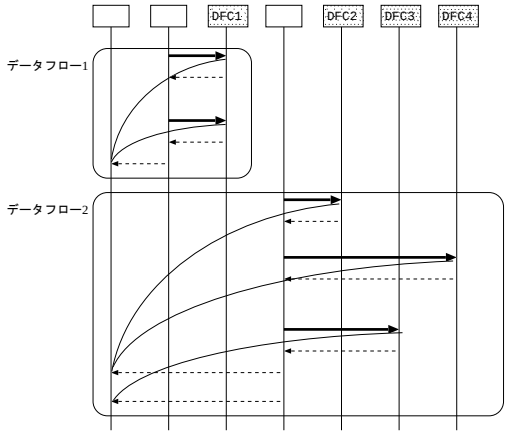

図 5: 重複部分マージ後のデータフロー範囲

5　適用実験

本節では，提案手法の有効性を評価するために，実際に運用されているプログラムに対して適用実験を行い，その結果を述べる．

5.1　実験の概要

提案手法の有効性を評価するために，実際に運用されているプログラムに対してトランザクションファンクションを抽出し，設計仕様書に記載されているトランザクションファンクションとの比較によって評価を行った．具体的には，設計仕様書と実行履歴取得時にシステムに対して行った処理をもとに，実際に動作しているトランザクションファンクションをあらかじめ測定しておき，提案手法によって抽出したトランザクションファンクションと比較した．

対象として 2 個のプログラムを使用した．1 個目は，ある企業で開発されたソフトウェア開発部品の貸出管理を行う Web アプリケーションプログラムである．以降，この対象プログラムをツール管理システムと呼ぶ．

2 個目は学生のグループによって開発された，図書管理システムである．4 組の学生グループが，同じ設計仕様書を基に作成したプログラム 4 個に対して提案手法を適用した．

5.2　実験の評価方法

実行履歴取得時に設計仕様書とプログラムの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，メソッド呼び出し群の範囲として取得しておく．メソッド呼び出し群の範囲と，提案手法によって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，該当する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．

図 6 に示すように，データフローがメソッド呼び出し群の範囲に含まれる場合に，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みなす．なお，データフローの終端点がメソッド呼び出し群の範囲外であっても，次のメソッド呼び出し群の範囲に含まれていなければ，正しく抽出できていると判定する．これは，取得したデータを出力結果として画面に出力する際に，画面表示を行うメソッドまでデータフローが及んでいるために起こる現象であり，トランザクションファンクションに対応するのメソッド呼び出し群において，トランザクションファンクションは正常に実行されていると考えられるからである．

出力されたデータフローが，あるデータフローの範囲に含まれてしまう場合，提案手法のアルゴリズムではマージが実行され，含まれる側のデータフローは出力されない．しかし，

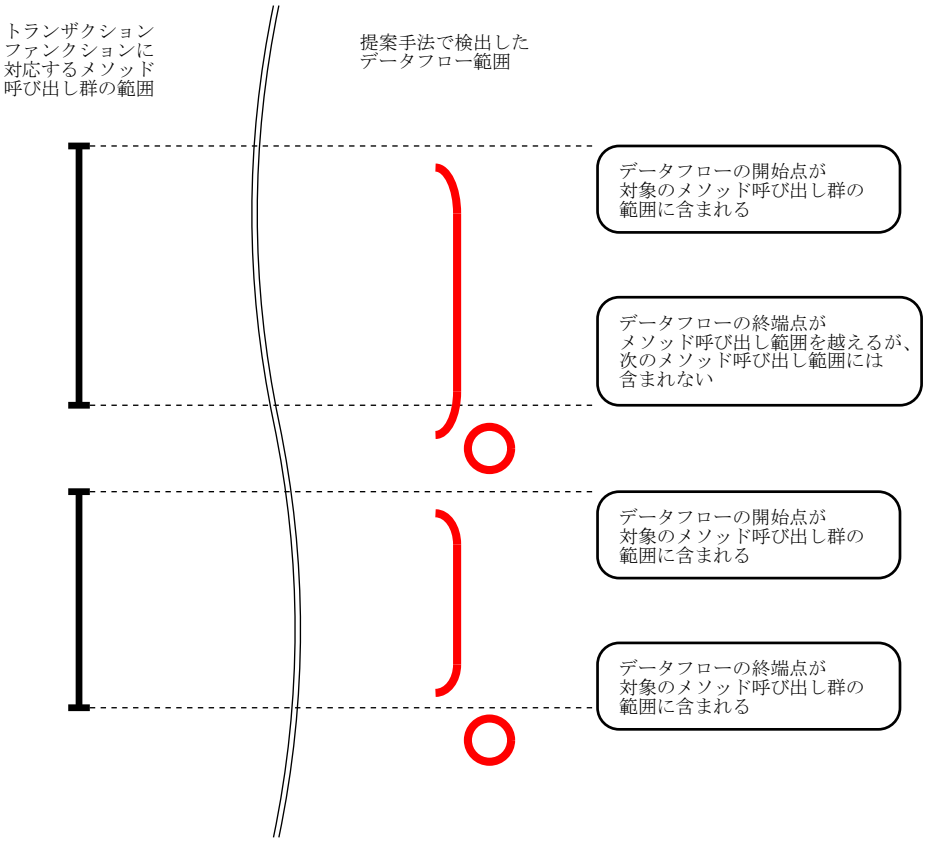

図 6: トランザクションファンクションを正しく検出する場合

包含する側のデータフローがメソッド呼び出し群と 1 対 1 で対応する場合には，図 6 と同様にトランザクションファンクションを正しく抽出できたと判定する．

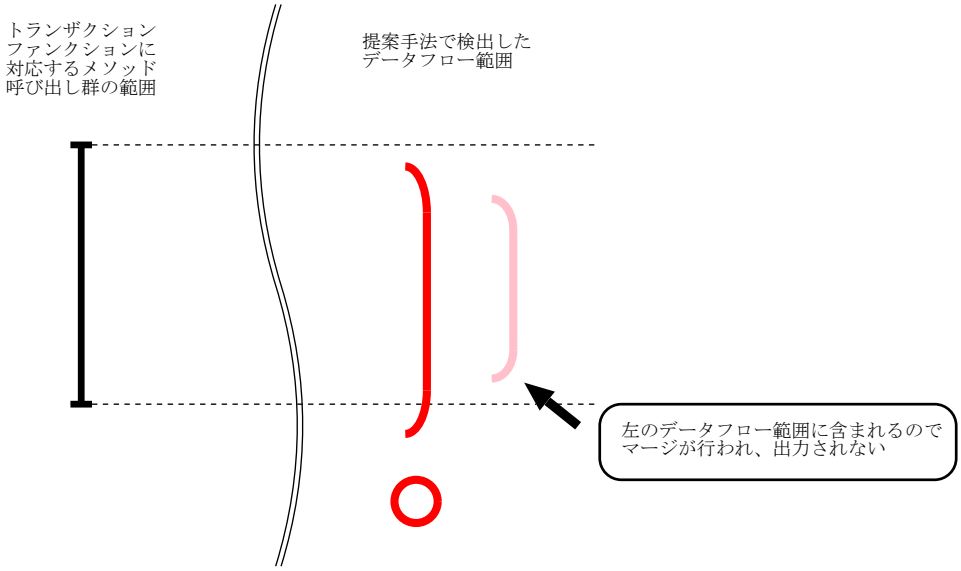

図 7: データフロー範囲のマージと抽出結果

次に，トランザクションファンクションが正しく検出できない場合について述べる．データフロー範囲の終端点が，対象のメソッド呼び出し群の範囲だけでなく，次のメソッド呼び出し群の範囲にまで及んでしまう場合は，トランザクションファンクションを正しく検出できないと判定する．図 8 に例を示す．

なお，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．

5.3　ツール管理システム

ツール管理システムの規模は約 37000LOC であり，実装されている機能は一般ユーザ向けとシステム管理者向けに分類されている．システム管理者はシステム管理者向けの機能だけでなく，一般ユーザ向けの機能も実行することができる．

一般ユーザ向けの機能

- ログイン
  ツール起動時に表れるログイン画面においてログインする．

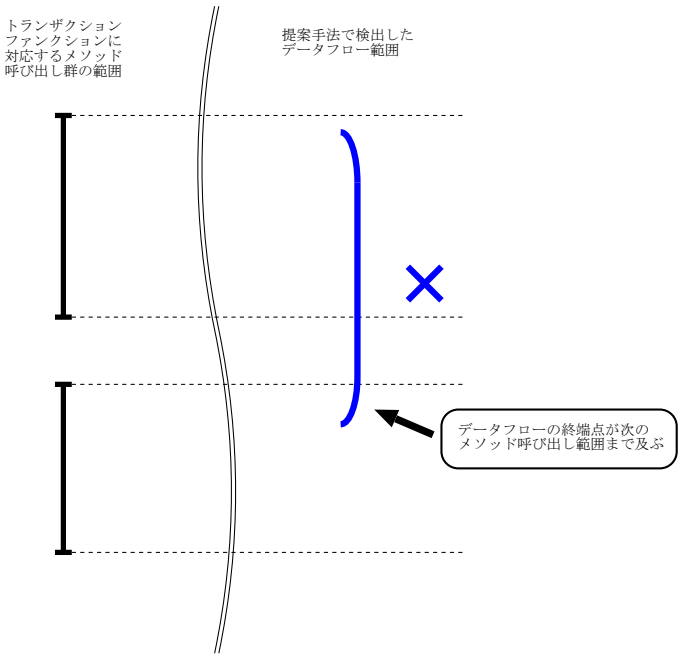

図 8: トランザクションファンクションを誤検出する場合

- 保有ツールの検索
  ログイン後の画面で検索条件を設定し保有ツールを検索すると，検索条件に当てはまり，現在利用可能なツールの一覧が出力される．そして，ツールの名前にはハイパーリンクを設定し，クリックすることで詳細な情報を見ることができる．検索結果が複数ある場合は，ソートや次のページの表示などを行える．

- ツールの貸し出し申請
  ツールの詳細な情報を表示した画面からツール貸し出し申請を行うと，申請画面が表示される．必須項目入力後，確認画面を経て貸し出し申請を完了する．

- ツールの新規登録
  新しく入荷したツールをアプリケーションによる管理下に加えるために登録処理を行う．ツールの資産番号や物品名称，備考情報などを記入し登録する．

- 備考メンテナンス
  ツールに付加してある備考情報の内容を変更する．また，備考記載欄の追加も実行できる．

システム管理者限定の機能

- 貸し出し申請の審査
  申請一覧表示画面から審査する案件の詳細を確認し，貸し出しを許可する場合は数量情報，返却先宛先，利用開始実績日を入力して貸し出しを実行する．数量情報には予約数と貸出数が保有数の合計になるような値を入力する．利用開始実績日には原則として当日のみ入力できる．また，申請の却下や削除をすることもできる．

- 返却処理
  返却申請のあった案件の詳細情報を表示させ，利用終了実績日に終了日を入力することで返却処理を受け付ける．

- 貸し出し履歴参照
  これまで行ったツール貸し出し履歴を表示させることができる．検索条件を付加して再検索を実行すると，条件に適合したもののみが表示される．条件をキーとするソートも可能である．

- ツールの登録審査
  ツールの新規登録申請が行われた案件に対して，承認，却下，詳細表示が実行できる．

提案手法を実験対象ソフトウェアに適用するために，データファンクションクラスを指定する必要がある．今回の適用実験では，設計仕様書において実際にデータファンクションとしてカウントされているテーブルとデータ項目をフィールド値としてもっているクラスを，データファンクションクラスとして 8 個指定した．

次節以降で，いくつかの実行履歴に対して提案手法を適用した際のトランザクションファンクション抽出結果と，設計仕様書と実行履歴から判断したトランザクションファンクションの比較を行い，考察する．

5.3.1 無作為に選んだ機能に対する実験

使用する実行履歴

無作為に選んだ機能を実行した際の実行履歴を 4 個用意し，それぞれに対して提案手法を適用した．以降では用意した実行履歴を，それぞれ履歴 1〜4 と記述する．以下に，各実行履歴取得時にツール貸出システムに対して行った処理を時系列で表す．

履歴 1

1. システムにログイン

2. 貸出可能な物品を検索
3. ある特定の物品の詳細情報を表示

履歴 2

1. システムにログイン
2. システムに登録してある物品の管理画面を表示
3. 登録物品の備考情報を画面に表示
4. システムからログアウト

履歴 3

1. システムにログイン
2. システムに登録してある物品の管理画面を表示
3. 登録物品の備考情報を画面に表示
4. ある物品の備考情報を変更
5. システムからログアウト

履歴 4

1. システムにログイン
2. システムに登録してある物品の管理画面を表示
3. 登録物品の備考情報を画面に表示
4. 備考情報表示欄を 1 行追加
5. 備考情報表示欄を 1 行削除して元に戻す
6. システムからログアウト

履歴 1〜4 には次の表 7，8，9，10 に示すトランザクションファンクションが含まれている．

表 7: 履歴 1 に含まれるトランザクションファンクション

利用中物品検索
物品検索
物品詳細

表 8: 履歴 2 に含まれるトランザクションファンクション

利用中物品検索
物品登録未承認案件一覧
物品管理品一覧
備考表示

履歴 2 と履歴 4 はツール貸出システムに対して異なる処理を行っているが，含まれるトランザクションファンクションは同一である．

実験結果

履歴 1〜4 の全てのテストケースにおいて，トランザクションファンクションを正しく抽出することができた．

無作為に取り出した機能に含まれるトランザクションファンクションが正しく抽出できたので，次節ではユーザが実行できる全ての機能を実行した際の実行履歴に対する適用実験の概要とその結果について述べる．

5.3.2 ユーザが実行できる全ての機能に対する実験

使用する実行履歴

ユーザが実行できる全ての機能を実行した際の実行履歴を取得し，提案手法を適用した．以下に，今回用いた実行履歴を取得する際にツール貸出システムに対して行った処理を時系列で表す．

実行履歴取得時にシステムに実行させた処理

1. システムにログイン
2. 貸出可能な物品を検索

表 9: 履歴 3 に含まれるトランザクションファンクション

利用中物品検索
物品登録未承認案件一覧
物品管理品一覧
備考表示
備考更新

表 10: 履歴 4 に含まれるトランザクションファンクション

利用中物品検索
物品登録未承認案件一覧
物品管理品一覧
備考表示

3. ある特定の物品の詳細情報を表示
4. 物品の貸出
5. システムに登録してある物品の管理画面を表示
6. 物品をシステムに新規登録
7. 物品の管理画面を表示
8. Top 画面 (ログイン時の画面) を表示
9. 利用中の物品を条件付きで検索
10. 物品の管理画面を表示
11. 登録物品の備考情報を画面に表示
12. ある物品の備考情報を変更
13. 備考情報表示欄を 1 行追加
14. 備考情報表示欄を 1 行削除して元に戻す
15. 物品の管理画面を表示

16. 管理者に承認される前の新規登録案件の物品情報を表示

17. 物品の管理画面を表示

18. システムが管理している任意の物品 1 個の詳細情報を表示

19. 表示した物品データに記されている特記事項を変更

20. システムが管理している任意の物品 1 個の詳細情報を表示

21. 表示した物品と同じ物品をシステムに追加登録 (利用できる物品の個数を増やす)

22. システムが管理している任意の物品 1 個の詳細情報を表示

23. 物品の管理画面を表示

24. システムからログアウト

この実行履歴には，表 11 に記述した 16 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．なお，実行履歴中には重複も含めてトランザクションファンクションは 35 回実行されている．

実験結果

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着目して，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．トランザクションファンクション抽出結果を表 12 に示す．

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」のような，ユーザにデータの詳細を与えるトランザクションファンクションを正しく検出することができなかった．また，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」も正しく検出できなかった．この 3 つのトランザクションファンクションは，画面に表示する選択肢の項目を検索，および取得する機能を持っている．図 9 は，ツール管理システムの画面をキャプチャした画像であり，アイテムグループ検索，コードテーブル検索の各トランザクションファンクションによって検索および取得されるデータ項目を示す．

次節では，同一の設計仕様書で開発された 4 個の図書管理システムに対して行った実験について説明する．

表 11: 実行履歴に含まれるトランザクションファンクション

利用中物品検索
利用中物品検索 (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

5.4 図書管理システム

図書管理システムは，4 グループの学生が，同一の設計仕様書に基づき，それぞれ実装を行ったシステムである．4 個のシステムを対象に実験を行い，同一の設計仕様書に基づいて異なる実装が行われた複数のシステムに対して提案手法を適用し，全てのシステムに対して同じ結果が得られることを検証する．

以降，学生の 4 グループをグループ A,B,C,D，各グループで実装されたシステムをそれぞれシステム A,B,C,D と記述する．

図書管理システムに実装されている機能を以下に示す．

実装されている機能

- ログイン
  ツール起動時に表示されるログイン画面からログインする．

- 新規ユーザの登録

表 12: 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	○
利用中物品検索 (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	×

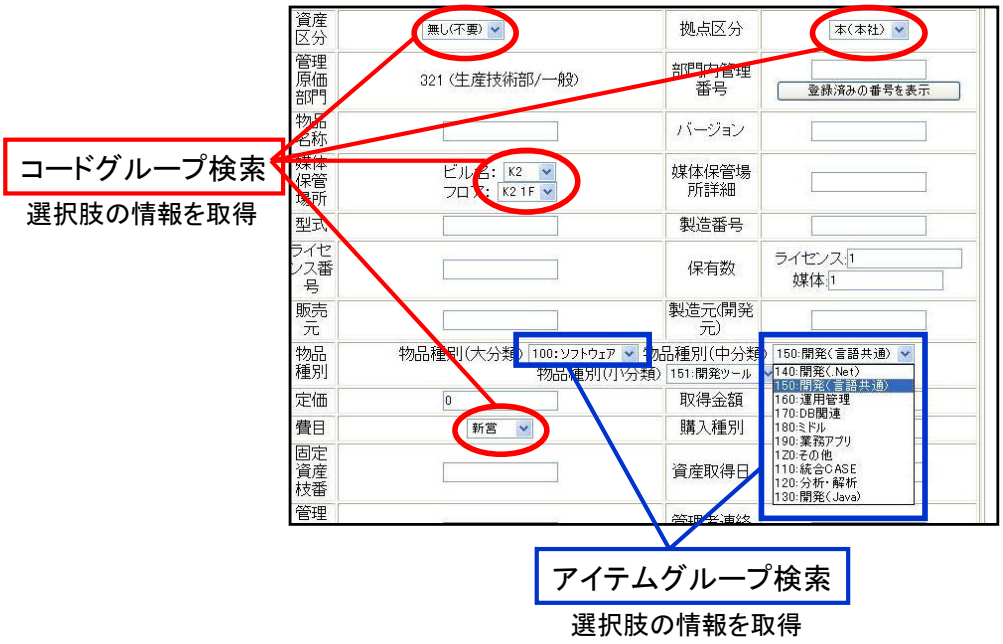

図 9: アイテムグループ検索・コードテーブル検索

ログイン画面内の「新規ユーザ」項目にチェックを入れ，登録したいユーザ名とパスワードを入力してログインすることによって，新規ユーザの登録が行われる．

- ツールが管理する図書の一覧を表示
  図書管理システムが管理している全ての図書を画面に表示する．各図書にはハイパーリンクが設定してあり，ハイパーリンクをクリックすることで図書の詳細情報を見ることができる．

- 図書の詳細情報を表示
  指定した図書のタイトル，ISBN として登録された 10 桁の番号を画面に表示する．図書が貸出可能の場合は画面内に「貸出ボタン」を表示させる．他のユーザに貸出中の場合は対象の図書を借りているユーザ名を表示する．自分が借りている図書の場合，画面内に「返却ボタン」を表示させる．

- 図書の貸出
  図書の詳細情報表示画面で,「貸出ボタン」を押すことによって図書の貸出を行う．

- 図書の返却
  図書の詳細情報表示画面で,「返却ボタン」を押すことによって借りている図書の返却を行う．

- 貸出中の図書を表示
  ログインしているユーザが借りている図書を画面に一覧表示する．図書一覧の表示と同様に，各図書にはハイパーリンクが設定してあり，クリックによって図書の詳細情報画面への遷移が行われる．

- 図書の新規登録
  新たな図書を図書管理システムの管理下に加える．図書追加の際には，図書のタイトルと ISBN，図書を保管するラックを指定する．ラック名は画面に表示される選択肢から選択する．

- 図書ウォッチ
  図書のウォッチとは，他のユーザに貸し出されている図書をウォッチリストに加え，図書が返却されたかどうかをチェックすることができる機能である．他ユーザに貸し出されている各図書に対してウォッチの開始・解除を設定することができ，ウォッチしている図書はウォッチリストに格納される．ウォッチリストを表示させることでウォッチしている全ての図書の貸出状況をチェックできる．

データファンクションクラスには，フィールドにデータ項目を保持し，各データにアクセスする際は getter,setter メソッドを利用するように実装されているクラスを，2 個指定した．

次節では，同一の設計仕様書に基づいて上で述べた機能を実装した，4 個の図書管理システムに対して，ユーザが実行できる全ての機能を実行した際の実行履歴に対する適用実験の概要と，その結果について述べる．

5.4.1 全ての機能に対する実験

使用する実行履歴

ユーザが実行できる全ての機能を実行して履歴を取得し，提案手法を適用した．以下に，実験で用いた実行履歴を取得する際に，4 個の図書管理システムに対して行った処理を時系列で表す．

図書管理システムに実行させた処理

1. システムにログイン
2. ウォッチリストを表示
3. 図書を新規登録
4. 管理している図書の一覧を表示

5. 図書を 1 個選び，詳細を表示
6. 詳細を表示した図書を借りる
7. 再度図書の詳細を表示
8. 借りている図書を返却
9. 他ユーザに貸し出されている図書のウォッチを開始
10. ウォッチしている図書の中で 1 冊の図書に対するウォッチを解除
11. ログインしているユーザが借りている図書を表示
12. システムからログアウト
13. 新規ユーザでシステムにログイン
14. システムからログアウト

この実行履歴には，次の表 13 に記述した 12 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．実行履歴中，重複も含めてトランザクションファンクションは 21 回実行されている．

表 13: 図書管理システムの実行履歴に含まれるトランザクションファンクション

ログイン
新規ユーザ登録
図書一覧表示
ウォッチリスト表示
貸出中図書表示
図書詳細表示
ラック一覧照会
図書追加
図書貸出
図書返却
図書ウォッチ開始
図書ウォッチ解除

実験結果

各グループのシステムに提案手法を適用して得られた実験結果を次の表 14 に示す.

表 14: 各システムで得られた実験結果

	システム A	システム B	システム C	システム D
履歴中の TF 出現回数 (重複含む)	21	21	21	21
検出したデータフロー数	89	97	105	109
TF に対応するデータフロー数	12	12	12	12

4 個のシステム A,B,C,D の全てにおいて，検出したデータフロー範囲のうち，12 個のデータフロー範囲がトランザクションファンクション実行部分と 1 対 1 で対応していた．トランザクションファンクションと対応していなかったデータフロー範囲は,「データファンクションクラスのデータ項目から読み出した URL を使って画像を表示する」という機能を実行している箇所であった．この画像表示機能は，仕様上，トランザクションファンクションとして識別されていないが，本手法では「システムに実装されたひとつの機能」として抽出されたと考えられる．

次に，各トランザクションファンクションに着目して，データフロー範囲との関係を示す．含まれる 12 種類のトランザクションファンクションのうち，システム A,B,C,D の全てにおいて,「図書一覧表示」・「ウォッチリスト表示」・「貸出中図書表示」・「図書詳細表示」・「図書追加」の 5 種を正確に抽出することができた．トランザクションファンクション抽出結果を,表 15 に示す.

5.5 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 メソッドの利用を推奨する，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．

データファンクションの実装が、本手法の想定するコーティング規約に対応していないために、図書管理システムにおいて、トランザクションファンクション「ログイン」・「新規ユー

表 15: 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
ログイン	×
新規ユーザ登録	×
図書一覧表示	○
ウォッチリスト表示	○
貸出中図書表示	○
図書詳細表示	○
ラック一覧照会	×
図書追加	○
図書貸出	×
図書返却	×
図書ウォッチ開始	×
図書ウォッチ解除	×

ザ登録」・「ラック一覧照会」・「図書貸出」・「図書返却」・「図書ウォッチ開始」・「図書ウォッチ解除」が検出できなかった。トランザクションファンクション「ログイン」・「新規ユーザ登録」が検出できなかったのは、各トランザクションファンクションで利用されるデータファンクションに関して、対応するデータファンクションクラスが存在しないことが原因である。上記の問題は、各システムのデータファンクションの実装が、対応するクラスによって実装されることを前提とした提案手法の想定と異なっているために生じた。

また，トランザクションファンクション「ラック一覧照会」・「図書貸出」・「図書返却」・「図書ウォッチ開始」・「図書ウォッチ解除」が検出できないのは，データ項目がデータファンクションクラスを経由せずに処理されるためである．この 5 種のトランザクションファンクションで利用されるデータ項目は，データファンクションクラスを経由せず，データベースから標準出力にデータが直接遷移するような実装が施されていた．提案手法で検出するデータフローは，DFC のフィールドのデータ依存関係を対象にしているため，データファンクションクラスを使用しないトランザクションファンクションは検出できない．

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，5.3.2 節の実験において，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から，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，ツール管理システムにおける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」は，5.3.2 節の実験では抽出できなかったが，節において行った，無作為に選んだ機能を実行した際の実行履歴を用いた実験では正しく抽出できている．図書管理システムに含まれるトランザクションファンクション「図書一覧表示」に対しても同様の現象が起きている．このことから，実行履歴の取り方によって抽出できるトランザクションファンクションの精度が変わるということがいえる．したがって，実行履歴を取得する際は，対象のシステムの設計や実装方法に関して，ある程度知識をもつ者が監修し，トランザクションファンクションが実行履歴に正しく反映されるような処理を実行させる必要がある．

さらに，ツール管理システムの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」のように，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，重複するデータフローをマージする際に条件を付加するという方法が挙げられる．同時に実行されるトランザクションファンクションを識別するための条件を与え，条件を満たすデータフローはマージせずに，それぞれトランザクションファンクションとみなす処理を実装することによって，同時に実行されるトランザクションファンクションを抽出できる．この問題点の解決には，マージ条件も含めて検討および試行することが今後の課題である．

6　関連研究

Sneed[11] 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．また、Fetcke ら [12] はユースケースモデルからファンクションポイントを測定する手法を提案している。適用実験によって提案した測定手法の有用性を示し、Object-Oriented Software Engineering (OOSE)[13] とファンクションポイントの関連について述べている。

本研究では、データファンクションとクラス、メソッド群とトランザクションファンクションをそれぞれ対応させているが、これとは異なる対応付けを行う場合がある。Whitmire[14] は、各クラスを論理ファイルに、システムの境界を越えたメッセージの送受信をトランザクションファンクションにそれぞれ対応させ、ファンクションポイントの計測を試みている。一方 Schooneveldt[15] らは、クラスをファイルに、オブジェクトによるクライアントへの service をトランザクションファンクションにそれぞれ対応させている。

ソフトウェアの規模を測定する尺度をファンクションポイント以外に新たに定義する研究も行われている。Antoniol ら [9] 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，構成されるクラスの組み合わせによって論理ファイルを 4 種類に分類し，4 種類全てについて OOFP 値を算出し、比較を行っている．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．トランザクションファンクションは複数メソッドの処理の組み合わせによって実装されることが多いため，文献 [9] と比べて本研究の提案手法は、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．文献 [16] ではオブジェクト指向ソフトウェアの規模を測定するために、“ object points ”という指標を定義している。object points は、対象ソフトウェアのクラス構造に着目して規模を

測定する手法であり、クラス構造以外にもメッセージ、プロセス、ユースケースを規模測定のパラメータに挙げている。文献 [17] では、“ Predictive Object Points (POPs) ”を考案している。POPs はクラスとクラスごとに重み付けされたメソッドの数に基づいて計測され、継承木の深さを調整要因として捉えて最終的な値を算出する。メソッドはその処理内容によって 5 つのタイプ (constructor, destructor, modifier, selector, iterator) に分けられ、さらに 3 段階 (高・中・低) の重み付けが行われる。

7 まとめ

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項目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みなして抽出を行い，実プログラムへの適用実験で提案手法の妥当性を確かめた．

今後の課題としては，以下の事柄が挙げられる．

- 重複データフローの処理

  現在は重複するデータフローを単純にマージしているが，これでは複数トランザクションが同時に実行された場合に誤ったトランザクションファンクション抽出結果を出力してしまう．この問題を解決するために，重複したデータフローが検出された場合には，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．

- データファンクションクラスの利用方法を再検討

  データファンクションクラスを経由せずに利用されるデータ項目の処理を，トランザクションファンクションとして識別できるように拡張する．また，データファンクションクラスとしてクラスを指定する方法を再検討し，データファンクションクラスを過不足なく指定できるようにする．

- 他の適用対象プログラムにおける実験

  他プログラムへの適用実験を行うことで，手法の汎用性を確立する．データフローを実行履歴上にマッピングする段階で，適用対象プログラムのソースコードの生成規約に基づいて，提案手法に調整を施す必要がある．

- ファンクションポイント値の計測

  提案手法を用いたトランザクションファンクション抽出結果をもとにファンクションポイント値を算出し，設計仕様書から測定したファンクションポイント値と比較を行う．

謝辞

本研究の全過程を通して，常に適切な御指導および御助言を賜り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 井上 克郎 教授に心より深く感謝いたします．

本研究の全過程を通して，常に適切な御指導および御助言を賜り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 教授に心から深く感謝いたします．

本論文を作成するにあたり，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 松下 誠 助教授に心から感謝いたします．

本研究を通して，逐次適切なご指導およびご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 谷口 考治氏に深く感謝いたします．

本研究に対して，開発現場の視点から貴重なコメントを頂いた，株式会社日立システムアンドサービス英繁雄氏，芝元俊久氏，前田憲一氏，津田道夫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情報科学研究科コンピュータサイエンス専攻井上研究室の皆様に深く感謝いたします．

参考文献

[1] 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.

[2] A.J.Albrecht: “Function Point Analysis”, in Jhon J.Marciniak, editor, Encyclopedia of Software Engineering, Vol.1, John Wiley & Sons, pp.518-524, 1994.

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

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

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

[6] 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.

[7] 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.

[8] B.A.Kitchenham: “The Problem with Function Points”, IEEE Software, Vol.14, No.2, pp.29-31, March/April, 1997.

[9] 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.

[10] 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.

[11] 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.

[12] Fetcke, T. Abran, A., and Tho-Hau Nguyen: “Mapping the OO-Jacobson approach into function point analysis”, Technology of Object-Oriented Languages and Systems, Proceedings, pp.192-202, July-Aug, 1997.

[13] Jacobson, I., M. Christerson, et al.: “Object-Oriented Software Engineering. A Use Case Driven Approach”, Addison-Wesley, 1992.

[14] Whitmire,S.: “Applying function points to object oriented software”, Software Engineering Productivity Handbook (J. Keyes, ed.), McGraw-Hill, pp229-244, 1993.

[15] Schooneveldt,M., Hastings,T., Mocek,J., Fountain,R.: “Measuring the size of object-oriented systems”, Proc. 2nd Australian Conference on Software Metrics, Australian Software Metrics Association, pp83-93, 1995.

[16] Harry M.Sneed: “ Estimating the costs of object-oriented software ”, Proc. of Software Cost Estimation Seminar, Systems Engineering Ltd., Durham, UK, 1995.

[17] Mehler,H., and Minkiewicz,A.: “Measuring object-oriented software with predictive object points”, Proc. ASM’97 - Applications in Software Measurement, Atlanta, 1997.