テクニカルノート

潜在的意味解析手法を用いたソフトウェア変更情報のクラスタリング手法

早瀬康裕† 今枝誉明† 市井誠†
松下誠† 井上克郎†

版管理システムのリポジトリ閲覧システムを用いることで，ソフトウェア開発者は過去にソフトウェアに対して行われた変更を参考にすることができる．しかし，既存のリポジトリ閲覧システムでは，同時に参照すべき変更を見落とす危険があった．そこで，本研究では，開発者に関連する変更を同時に提示することを目的とした，変更内容に基づいた変更のクラスタリング手法を提案する．また，提案手法を実現するシステムを試作し，実際のプロジェクトデータに対して適用した結果，本手法の有効性と問題点を確認することができた．

An Approach to Clustering Software Changes Using Latent Semantic Analysis

Yasuhiro Hayase,† Takaaki Imaeda,† Makoto Ichii,†
Makoto Matsushita† and Katsuro Inoue†

Repository browsers enable software developers to refer to past changes of software. However, developers sometimes miss the changes that should be referred to at the same time. In this paper, a method of clustering software changes is proposed for providing the developers related changes. Effectiveness and problems of the method are confirmed by applying the method to actual project data.

1. はじめに

近年，複雑化しているソフトウェアを効率良く開発し管理するために版管理システムが広く用いられている[1)]．版管理システムとは，ソフトウェアを構成するファイルであるソースコードやドキュメント等に対する変更を記録するシステムである．

版管理システムに蓄積された過去の変更を理解することで，効率的な新規開発や保守作業を行うことができるといわれている[2)]．具体的には，過去の類似した変更を真似ることで，機能追加や欠陥修正を簡単に行うことができる．

しかし，過去に行われた個々の変更を調べるだけでは，関連する変更を見落とし，誤解が生じる可能性がある．たとえば，ある箇所の変更の影響により他の箇所が変更されていた場合や，ある変更で欠陥を混入させ，後に行われた変更でその欠陥が修正されていた場合等に，前者の変更のみを見て同様の変更を行うと欠陥を埋め込んでしまうことになる．

そのため，開発者は過去の変更を参考にする際には，すべての関連する変更を把握しなければならない．しかし，蓄積された膨大な変更の中から関連する変更を人手で探すのは困難であり，開発者の負担となっている．

そこで，本研究では変更箇所に含まれる単語を利用して，版管理システムに蓄積されたファイルに対する変更をクラスタリングする手法を提案する．また，提案手法を実現するシステムを試作して実際のソフトウェアに対して適用し，本手法により関連のある変更の集合が抽出できるかどうかを確認する．

2. 版管理システムを用いたソフトウェア開発

版管理システムとは，ファイルに対して施された追加・削除・修正等の変更を記録するシステムである．版管理システムにはリポジトリと呼ばれるデータベー

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

スが存在し，リポジトリには管理対象となるすべてのファイルの，過去の任意の時点での状態が保存される．過去のある時点でのファイルの状態はリビジョンと呼ばれ，時系列に従って連続した番号（リビジョン番号）で識別される．

版管理システムを用いた開発は以下のような手順で行う．まず，開発者はリポジトリから必要なファイルを手元にコピーし，変更を加える．そして変更したファイルをリポジトリに登録（コミット）することで，ファイルの新たなリビジョンが生成される．コミットは，複数のファイルに対して同時に行うことができる．この一連のコミットをコミットトランザクション（以下単にトランザクション）と呼ぶ．また，コミットする際に，変更理由等を記述した文章（コミットログ）を残すことができる．

3. 変更情報のクラスタリング手法

1 章で述べた問題点を解決するために，関連する変更をまとめて開発者に提示する手法を提案する．本手法では，隣接するリビジョン間の差分とコミットログの中に出現する単語に着目し，この単語を利用したクラスタリングを行う．クラスタリングに必要な変更の関連を求めるために，潜在的意味解析[3]（以下 LSA）を用いる．LSA は自然言語で記述された文書間の類似度を，単語間の潜在的な関連を考慮して算出する手法である

提案する手法の概要を**図 1** に示す．本手法は以下の 3 つのステップからなる．

(1) リポジトリからのトランザクション情報の抽出
(2) LSA を用いたトランザクション情報間の類似度計算
(3) 類似度に基づくトランザクション情報のクラスタリング

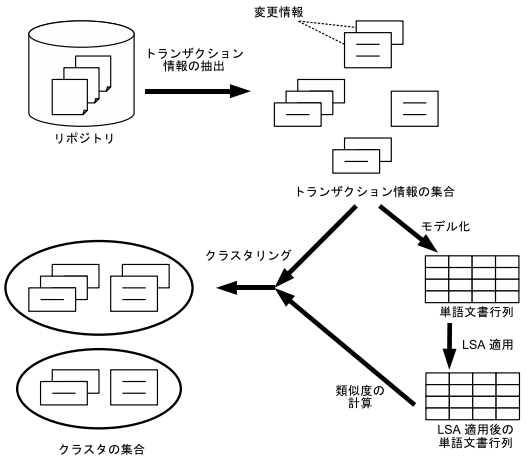

図 1 手法の概要
Fig. 1 Overview of the approach.

以降，各ステップの詳細を説明する．

3.1 リポジトリからのトランザクション情報の抽出

手法の最初のステップでは，版管理システムのリポジトリから**変更情報**の集合（**トランザクション情報**）を取り出す．変更情報は以下の情報からなる．

- 変更したファイルのパス名
- 変更後のリビジョン番号
- コミットした開発者
- コミットされた日時
- コミットログ
- 変更差分

変更差分とは，直前のリビジョンとの行単位での差分，すなわち変更開始行番号と変更範囲，追加行の内容，削除行の内容である．

3.2 LSA を用いたトランザクション間の類似度計算

このステップでは，LSA を用いてトランザクション間の類似度を求める．

LSA はベクトル空間モデル[4] に基づいた手法であり，文書内に出現する各単語の出現回数を要素とする行列（単語文書行列）を入力とする．本手法では，トランザクション情報を文書に対応させる．そして，トランザクション情報 T 中の単語 w の出現数を，T のコミットログと T に含まれるすべての変更差分に w が出現した回数とする．

この行列に LSA を適用して変換する．変換された行列の文書ベクトル間の cos の値を求めることで，トランザクション間の類似度が求められる．

3.3 類似度に基づくトランザクション情報のクラスタリング

このステップでは，トランザクション情報間の類似度に基づいて，トランザクション情報をクラスタリングする．クラスタリングアルゴリズムには，階層的クラスタリング手法[5] を採用した．また，クラスタ間の類似度算出には，完全リンク法[5] を採用した．クラスタ c，d 間の類似度 $\mathrm{sim}(c,d)$ は，以下の定義を用いる．

$$\mathrm{sim}(c,d) = \min_{v_c \in c, v_d \in d} \cos(v_c, v_d)$$

具体的なクラスタリングのアルゴリズムは，以下のとおりである．ただし，s_{th} は閾値を表す．

入力：トランザクション情報 $T_1, T_2, \ldots, T_N$
（N：総トランザクション数）

出力：トランザクション情報のクラスタリング結果 S
初期状態：$S \leftarrow \{\{T_1\}, \{T_2\}, \ldots, \{T_N\}\}$
(1) 以下の条件を満たすクラスタの組 (a, b) ($a, b \in S$) を求める：
$a \neq b \ \wedge \ \forall c,\ d \in S,\ c \neq d \ \wedge \mathrm{sim}(a, b) \geq \mathrm{sim}(c, d)$
(2) $\mathrm{sim}(a, b) \leq s_{\mathrm{th}}$ ならば，クラスタリング終了
(3) $S \leftarrow S \cup \{a \cup b\} - \{a, b\}$
(4) (1) に戻る

4. 適用事例

本手法により関連のある変更情報が抽出できるか確認するため，本手法を実現するシステムを試作し，実際のソフトウェアのリポジトリに対して適用した．適用対象は，我々の研究室で開発されたソフトウェア部品検索システム SPARS-J[6] である．SPARS-J の概要は**表 1** のとおりである．

4.1 適用方針

クラスタリングの閾値 s_{th} は，経験に基づいて 0.8 とした．

クラスタリング結果を評価するために，生成されたクラスタのうち，複数のトランザクション情報を含むクラスタを以下の条件に基づいて手作業で分類する．

(1) クラスタ中に以下のような関係を含むトランザクションの組 (a, b) が存在する場合
  (1-a) a で作りこんだ欠陥を b が修正した．
  (1-b) a と同じ箇所を b がリファクタリングした．
  (1-c) a と b が同じ目的を持って行われた．
  (1-d) a の影響によって b が行われた．
  (1-e) a の変更を b が取り消した．
(2) (1) を満たさないが，何らかの関連を持つクラスタが含まれる．
(3) 関連のあるトランザクションを含まない．

上記 (1) に分類されたクラスタを，同時に参照することが望ましいトランザクションの集合であると考え，「有用である」と判断する．

次節で，この方針に基づいた適用結果を示す．

表 1 SPARS-J の概要
Table 1 Overview of SPARS-J.

開発期間	2003 年 3 月〜2006 年 1 月
総リビジョン数	2,795
総トランザクション数	834
ソースファイル数	345
最新リビジョンでの総行数	95,479
開発言語	C/C++

4.2 適用結果

クラスタリングの結果を**表 2** に示す．本手法により抽出されたクラスタのうち，有用だと判断されたクラスタは 55.2%存在した．

図 2 に，(1-a) に分類されたクラスタの一例を示す．このクラスタは 3 つのトランザクション情報からなる．この例では，データベースの操作時に必要なある関数に関連するトランザクションの集合が抽出できた．1 つ目のトランザクションでは，ファイル `SPARS/src/DB/db_common.c` で関数 `compare_asbigendian` を定義し，その中で変数 `ai` が `u_int8_t *` 型として宣言された．2 つ目のトランザクションでは，その変数に代入している箇所でキャストを行うよう修正されたが，これが間違っていたため 3 つ目のトランザクションで同一箇所がさらに修正された．

(2) に分類されたクラスタには，ソースコードの整形を行ったトランザクションや，マージを行ったトランザクションによりクラスタを形成していた例があった．

また，(3) に分類されたクラスタでは，含まれるト

表 2 SPARS-J を対象としたクラスタリング結果
Table 2 Result of clustering on SPARS-J.

(1)	64	(55.2%)
(1-a)	33	(28.4%)
(1-b)	10	(8.6%)
(1-c)	14	(12.1%)
(1-d)	2	(1.7%)
(1-e)	5	(4.3%)
(2)	28	(24.1%)
(3)	24	(20.7%)
計	116	(100%)

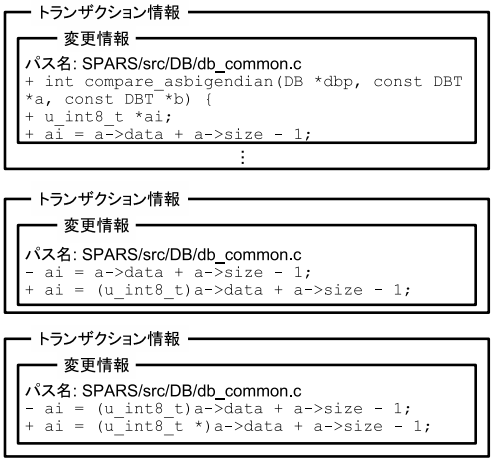

図 2 SPARS-J を対象としたクラスタの例
Fig. 2 An example of a cluster on SPARS-J.

ランザクション情報間で，コミットログやコメント文中の一般的な単語や，CGI プログラム中の HTML タグ等が共通して出現しており，クラスタを形成していた例があった．

4.3 考　　察

本稿の適用事例では，有用と思われるクラスタを 55.2%生成できた一方，含まれるトランザクション間に関連がないクラスタが 44.8%存在した．このことから，関連がないと考えられるクラスタを生成しないようにする必要があると考えられる．

このように，関連がないクラスタが生成された原因は，出現する単語をもとにする手法の性質によるものであると考えられる．生成されたクラスタを観察した結果，内容に以下のような傾向があることが分かった．まず，トランザクション情報間に関連のあるクラスタには，ソースコード中の変数や関数名等の識別子が共通して現れていた．一方，トランザクション情報間に関連のないクラスタでは，共通して現れる単語数が少なく，識別子以外の単語が共通して現れていた．

以上のことから，関連がないクラスタを減少させる方法として，単語の抽出時に構文解析を行うことで識別子以外の単語の抽出を抑制し，有用でないクラスタの生成を抑えることや，出現する単語やその数に基づいてクラスタをフィルタリングすることが考えられる．

また，本実験ではクラスタリングの閾値を経験に基づいて決定した．しかし，この値は対象とするソフトウェアのリポジトリや，抽出された単語の総数等によって変化するものと考えられるため，リポジトリ規模や単語数を考慮した閾値決定方法の確立も必要である．

5. 関 連 研 究

Zimmermann ら[7] は，版管理システムに蓄積された変更履歴から，ファイルや変数・関数等のプログラム要素間の相関ルールを抽出し，ある要素が変更された際に相関ルールをもとに変更すべき要素を提示する手法を提案した．我々の手法は，プログラム要素だけではなく，影響による具体的な変更内容まで特定することができるため，開発者の理解をより支援することができる．

Canfora ら[8] は，版管理システム中のコミットログと，欠陥追跡システム中の変更要求記述との文書上の類似度をあらかじめ計算しておき，新たな変更要求があったときに，過去の類似した変更要求を抽出し変更される可能性のあるファイルを提示する手法を提案した．我々の手法によって類似した変更を抽出することで，Canfora らの手法で直接発見することができなかったファイルを提示することが可能となると考えられる．

6. ま と め

本研究では，版管理システムに蓄積された変更情報をクラスタリングし，関連ある変更を抽出する手法を提案した．また，提案した手法に基づいてシステムを試作し，実際のソフトウェアに対し適用を試みた．その結果，関連のある変更をある程度抽出できることが分かった．

今後の課題としては，単語の適切な抽出手法やクラスタリングの閾値決定方法の確立，手法の定量的な評価等があげられる．さらに，システムを実用化に向けて，開発者の理解をより効率的に支援できる，クラスタの提示方法を考案する必要がある．

参 考 文 献

1) Estublier, J.: Software configuration management: A roadmap, *Proc. Conference on The Future of Software Engineering*, pp.279–289, ACM Press (2000).
2) Feiler, P.H.: Configuration Management Models in Commercial Environments, Technical report CMU/SEI-91-TR-7, Carnegie-Mellon University (1991).
3) Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W. and Harshman, R.A.: Indexing by Latent Semantic Analysis, *JASIS*, Vol.41, No.6, pp.391–407 (1990).
4) Salton, G. and McGill, M.J.: *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc. (1986).
5) 徳永健伸：情報検索と言語処理，東京大学出版会 (1999).
6) 横森励士，梅森文彰，西　秀雄，山本哲男，松下誠，楠本真二，井上克郎：Java ソフトウェア部品検索システム SPARS-J，信学論 (D-I)，Vol.J87-D-I, No.12, pp.1060–1068 (2004).
7) Zimmermann, T., Weisgerber, P., Diehl, S. and Zeller, A.: Mining Version Histories to Guide Software Changes, *Proc. 26th ICSE*, pp.563–572, IEEE Computer Society (2004).
8) Canfora, G. and Cerulo, L.: Impact Analysis by Mining Software and Change Request Repositories, *Proc. 11th METRICS*, p.29 (2005).

(平成 19 年 5 月 2 日受付)
(平成 19 年 7 月 3 日採録)

早瀬　康裕（正会員）

平成 14 年大阪大学基礎工学部情報科学科卒業．平成 19 年同大学大学院情報科学研究科博士後期課程修了．現在，同大学特任助教．博士（情報科学）．オープンソースソフトウェア開発，ソフトウェア保守の研究に従事．

今枝　誉明

平成 17 年大阪大学基礎工学部情報科学科卒業．平成 19 年同大学大学院情報科学研究科博士前期課程修了．

市井　　誠（学生会員）

平成 16 年大阪大学基礎工学部情報科学科卒業．現在，同大学大学院情報科学研究科博士後期課程在学中．ソフトウェア部品検索の研究に従事．

松下　　誠（正会員）

平成 5 年大阪大学基礎工学部情報工学科卒業．平成 10 年同大学大学院博士課程修了．同年同大学基礎工学部情報工学科助手．平成 14 年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助手．平成 17 年同専攻助教授．平成 19 年同専攻准教授．博士（工学）．ソフトウェアプロセス，オープンソースソフトウェア開発の研究に従事．

井上　克郎（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士課程修了．同年同大学基礎工学部情報工学科助手．昭和 59〜61 年ハワイ大学マノア校情報工学科助教授．平成 1 年大阪大学基礎工学部情報工学科講師．平成 3 年同学科助教授．平成 7 年同学科教授．平成 14 年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教授．工学博士．ソフトウェア工学の研究に従事．日本ソフトウェア科学会，電子情報通信学会，IEEE，ACM 各会員．