派生プロダクト群における要求・実装間のトレーサビリティリンク抽出

土屋 良介† 鷲崎 弘宜† 深澤 良彰†

あらまし 派生開発されてきたプロダクト群に対して，保守や拡張を行う際には，要求と実装の間でトレーサビリティが明確化されていることが望ましい．しかし，大規模なプロダクト群に対して，これらの作業を人手のみで行うことは困難である．我々は，構成管理ログを用いて，要求資産とソースコード，それぞれの共通性・可変性分析結果間の対応関係を分析し，その結果を利用することで，要求・実装間のトレーサビリティリンクを自動的に抽出する手法を提案する．提案手法を，一定規模の派生プロダクト群に適用した結果，実用的な時間内で，妥当なトレーサビリティリンクを抽出できた．

Extraction traceability links between requirements and implementation in the same series of software products

Ryosuke Tsuchiya† Hironori Washizaki† Yoshiaki Fukazawa†

Abstract When performing maintenance and expansion for the group of products derived, it is important to clarify traceability between requirements and implementation. However, for the large group of products, these tasks manually are difficult. First, we analyze commonality and variability in requirements and source codes. Next, using the configuration management log, we analyze the correspondence of those results. Finally, using the results of this study, we propose a method to extract traceability links between requirements and implements automatically. Result of applying the proposed method to derived products group of a certain size, we were able to extract the appropriate traceability links in a practical time.

1. はじめに

派生開発されてきたプロダクト群に対して，保守や拡張を行う際には，要求と実装の間でトレーサビリティが明確化されていることが望ましい．大規模な派生プロダクト群のトレーサビリティリンクを手動で抽出することは現実的でないので，自動的にリンクを抽出する手法が必要となる．

要求と実装の間で，抽象度や表現方法が異なる場合には，それらのギャップを埋める情報が必要となる．そこで我々は，要求と関連のある情報と，実装箇所の情報が含まれる構成管理ログを利用することで，大規模な派生プロダクト群のトレーサビリティリンクを抽出する手法を提案する．本研究では，ソフトウェア開発に際して，開発者が記述したメッセージと，追加・修正を行ったファイルのパスが，それぞれ記録されたリビジョンの集合を構成管理ログとして扱う．

構成管理ログを用いた抽出では，ファイルのパスを利用する為，要求・関数間の細かいリンクは抽出出来ない．これらのリンクを抽出する為に，我々は派生プロダクト群に対する共通性・可変性分析を利用した．共通性・可変性分析とは，プロダクトを構成する要素について，それぞれがどのプロダクトに属しているものなのかを分析することである．

以下に，本研究の研究課題を定義する．

RQ1 構成管理ログを持つ大規模な派生プロダクト群において，要求・実装間のトレーサビリティリンクを自動的に抽出できるか

RQ2 対象となる派生プロダクト群が大規模であっても，実用的な時間内に，要求・実装間のトレーサビリティリンクを抽出できるか

以上の 2 つの研究課題が，提案手法により解決可能か検証を行っていく．

2. 提案手法

本手法の適用対象は，同一組織内で作られた大規模な派生プロダクト群である．入力として，各プロダクトの要求資産とソースコード，構成管理ログを必要とする．抽出されるトレーサビリティリンクを精査する為に，リンク抽出に先だって，要求資産とソースコードに対して共通性・可変性分析を行う．したがって，提案手法を以

†早稲田大学情報理工学科 Dept. Computer Science, Waseda University 〒169-8555 東京都新宿区大久保 3-4-1

下の 4 つのステップに分けて設計した.

(1) ベクトル空間モデルを用いた要求資産の共通性・可変性分析
(2) コードクローン分析を用いたソースコードの共通性・可変性分析
(3) 構成管理ログを用いた要求・実装間のトレーサビリティリンクの抽出
(4) 共通性・可変性分析結果を用いたトレーサビリティリンクの精査

以上を含めた, 提案手法の全体像を図 1 に示す.

ステップ(1), (2)では, 派生プロダクト群に対する共通性・可変性分析を行う. 要求とコード要素(コンポーネントと関数)が, それぞれどのプロダクトに属しているか分析する. コンポーネントとは, 1つのファイルやクラスを表す. 分析手法に利用する要素技術としては, 既存研究でも利用されているベクトル空間モデルとコードクローン分析を用いる[1][2].

そして, ステップ(3)で, 要求資産・ソースコードそれぞれの共通性・可変性分析結果を入力として, 構成管理ログを利用することで, 要求・実装間のトレーサビリティリンクを抽出する. 要求1つ1つに複数の特徴語を設定し, それらの特徴語を利用して, 構成管理ログ内を検索し, 実装箇所を特定する. 特徴語は, TF-IDF を利用して自動的に抽出した候補を参考に, ユーザーが設定する.

最後に, ステップ(4)で, 抽出されたトレーサビリティリンクに対して, 要求とソースコード, それぞれの共通性・可変性分析結果を比較することでリンクの精査を行う. 共通性・可変性分析結果にズレが生じている場合, 要求・実装間の粒度のギャップや, 仕様書とソースコードの間の矛盾が示唆される.

3. 適用実験

本研究では, まず手法を確立することを目的に, 一定規模の派生プロダクト群を対象とした適用実験を行った. 実験対象として, C 言語のテスティングフレームワーク, CUnit[3]を選択した. CUnit のバージョンの内, ver2.0-1, ver2.1-0, ver2.1-2 の 3 つを対象として, 要求・実装間のトレーサビリティリンク抽出を行った. CUnit のソフトウェア規模は 6～8KLOC である. 実験には Java で実装したツールを用いた.

共通性・可変性分析の結果, 15 種の要求と 10 種のコンポーネントが抽出された. 構成管理ログを用いて, これらの要求とコンポーネントの間のトレーサビリティリンクを抽出した. ドキュメントを利用して手動で抽出したリンク 20 個の内, 15 個のリンクを提案手法で抽出できた. 5 個のリンクが抽出出来なかった原因としては, リビジョン数が少なかったことが考えられる.

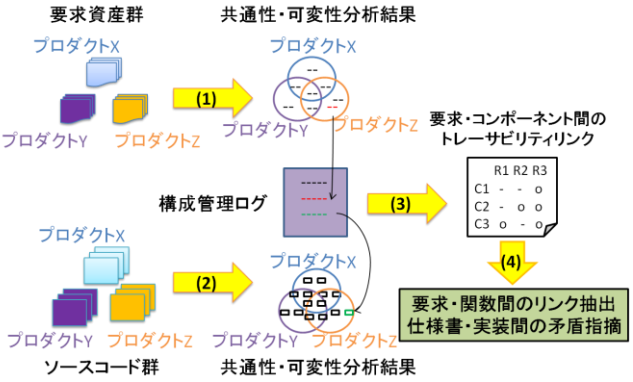

図 1 提案手法の全体像

ツールの実行時間は計 2 分であった. しかし, 構成管理ログ内を検索する際に用いる, 特徴語の設定には, 別途コストが必要となる.

要求とコンポーネントの共通性・可変性分析結果にズレが生じている 4 個のリンクについて, トレーサビリティリンクの精査を行った. これらは, ver2.1-2 固有の要求と, 3Ver 共通のコンポーネント群の間のリンクである. 関数の共通性・可変性分析結果を利用することで, これらのコンポーネントが持つ ver2.1-2 固有の関数を, 要求と結び付けた.

4. おわりに

我々は, 大規模な派生プロダクト群における, 要求・実装間のトレーサビリティリンクを自動的に抽出する手法を提案した. 構成管理ログと, 派生プロダクト群に対する共通性・可変性分析を利用した. 今後の課題として, 大規模な派生プロダクト群を対象とした適用実験及び, 属人的である特徴語設定手法の改善を目指す.

参考文献

[1] K. Kumaki, R. Tsuchiya, H. Washizaki and Y. Fukazawa, “Supporting Commonality and Variability Analysis of Requirements and Structural Models,” MAPLE 2012, SPLC’12, vol.2, pp.115-118, 2012.
[2] 吉村健太郎, Ganesan, D. and Muthig, D, “プロダクトライン導入に向けたレガシーソフトウェアの共通性・可変性分析法,” 情報処理学会論文誌, vol.48, no.8, pp.2482-2491, 2007.
[3] CUnit, http://sourceforge.net/projects/cunit/