

CENTRIS: A Precise and Scalable Approach for Identifying Modified Open-Source Software Reuse

Seunghoon Woo, Sunghan Park,
Seulbae Kim, Heejo Lee, Hakjoo Oh

発表者：西山 知弥

ICSE 2021

背景

- 適切な管理を行わずにOSSを再利用すると, ソフトウェアの保守性やセキュリティが損なわれる可能性がある
- ソフトウェア構成分析(SCA)を使用することで, ソフトウェア内でOSSコンポーネントがどのように再利用されているかを把握できる

*しかし既存のSCA技術は, 再利用されたOSSが変更されていないことを前提としているため, 変更された再利用を特定することが困難!

その他既存SCAに影響を与える要因

(1) 既存の OSS の一部だけを利用したり, ソースコードや構造の変更

(2) 再利用された OSS に複数のサブ OSS コンポーネントが含まれ, さらにサブ OSS コンポーネントには他の OSS コンポーネントが含まれる場合 ← 入れ子

(3) OSS プロジェクトの数は急速に増加しており, コードサイズも大きくなっていること [9]

[9] A. S. Barb, C. J. Neill, R. S. Sangwan, and M. J. Piovoso, "A statistical study of the relevance of lines of code measures in software projects," in *Innovations in Systems and Software Engineering*, vol. 10, no. 4. Springer, 2014, pp. 243–260.

目的

- SCA技術の精度向上
- OSSコンポーネントが入れ子になっている場合でも, 変更や修正されたOSSの再利用を正確に検出すること

アプローチ

- 新しいSCA技術であるCENTRISの提案
- OSSコードベースを分割し, OSSのユニークな部分のみの再利用を検出する

CENTRISの概要

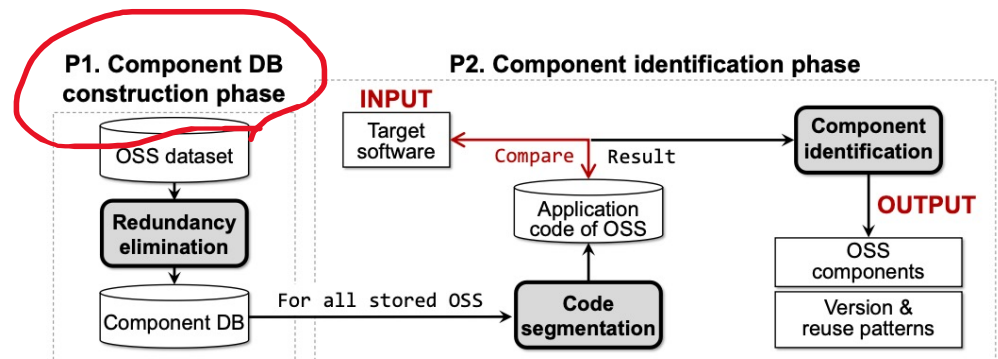


Fig. 2: High-level overview of the workflow of CENTRIS.

OSS プロジェクトを処理してコンポーネント DB を作成

- 複数のバージョンに共通する機能を一度だけ対象ソフトウェアと比較

(1) ある OSS のすべてのバージョンの関数を抽出する

(2) OSS の全バージョン数と同数のビンを作成する

関数を保存する際には, Locality Sensitive Hash (LSH) を適用 . LSH は 2 つのハッシュの類似性を測定する

- コンポーネント DB を構築

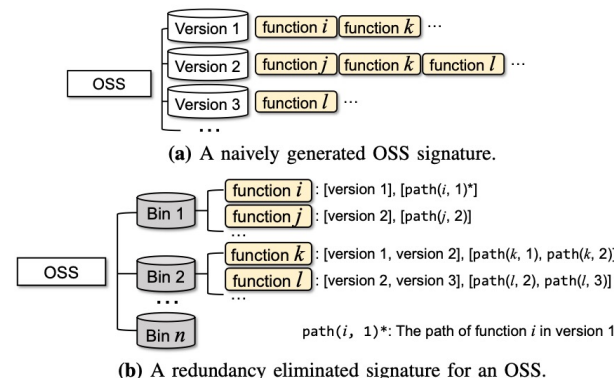


Fig. 3: Illustration of OSS signatures. We generate signatures for each OSS in the manner shown in (b), thereby reducing space complexity.

CENTRISの概要

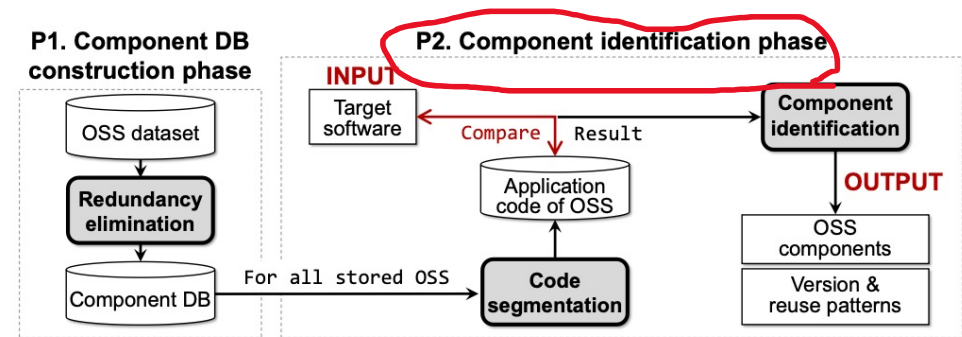
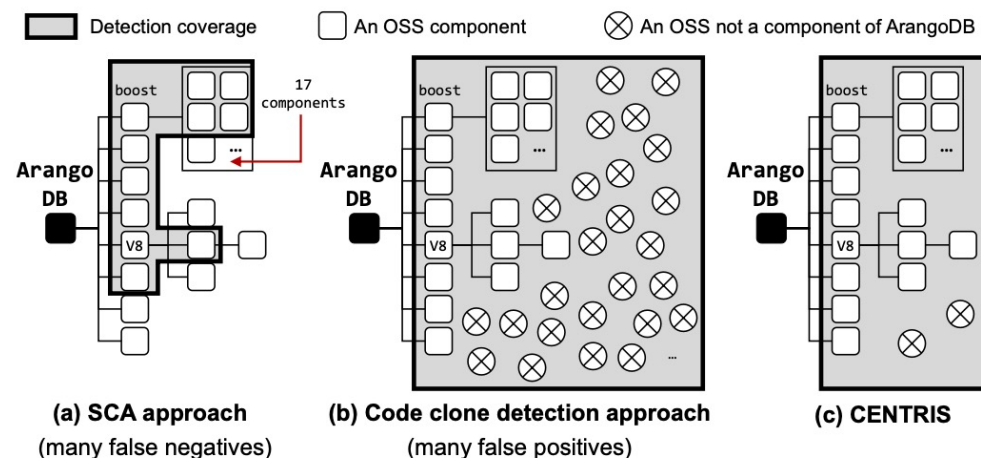


Fig. 2: High-level overview of the workflow of CENTRIS.

対象ソフトウェアで再利用されている OSS コンポーネントを特定

- コンポーネント識別処理とコードの細分化
 - (1) コンポーネントDBからプライムOSSを検出する
 - (2) すべてのOSSプロジェクトからアプリケーションコードを抽出
 - (3) ターゲットソフトウェア内のコンポーネントを特定
- 対象ソフトウェアの全関数を抽出した後, テキスト前処理とLSHアルゴリズムを適用して行われる

CENTRISのコンポーネント検出範囲



*対象はArangoDB

- 既存の SCA (OSSPolice [5]など)では,構造的な変更が行われたコンポーネントや,ごく一部しか再利用されていないコンポーネントを検出できなかった
- コードクローン検出アプローチ(Dé ja`Vu [11]など)では多数の誤検出が報告されている

[5] R. Duan, A. Bijlani, M. Xu, T. Kim, and W. Lee, "Identifying Open- Source License Violation and 1-day Security Risk at Large Scale," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 2169–2185.

[11] C. V. Lopes, P. Maj, P. Martins, V. Saini, D. Yang, J. Zitny, H. Sajjani, and J. Vitek, "Dé ja`Vu: a map of code duplicates on GitHub," in *Proceedings of the ACM on Programming Languages*, vol. 1, no. (OOPSLA). ACM, 2017, p. 84.

再利用パターンの分析

- GitHubの10,241のC/C++リポジトリから, 229,326のバージョンと800億のコードライン(LOC)からなる大規模なOSSデータセットを収集
- 各OSSの代表的なバージョン(最も機能が充実しているバージョン)を選んだ
- 選んだリポジトリの4,434(44%)が少なくとも1つのOSSを再利用しており, 計14,997のコンポーネントを検出

再利用パターンの分析

(1)自動検証 : 8,066件(54%)

検出された OSS の 3 つの要素(パス、ヘッダファイル、メタデータファイル)のうち、少なくとも 1 つが対象ソフトウェアに含まれていることを確認

(2) 手動による検証: 残りの6,931件(46%)

自動化された方法で検証できなかった 残りの結果については, 手動で検証を行なった

*2名で2週間かけて行なっただけ

OSSの再利用パターン

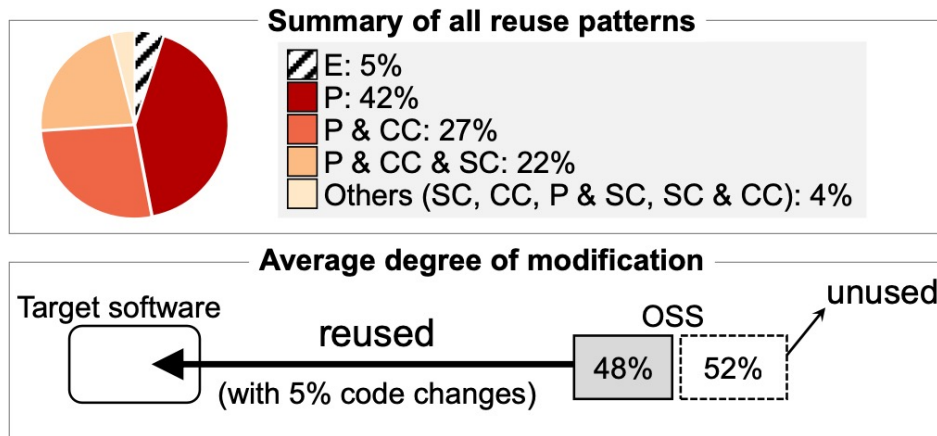
(1)完全再利用(E) : OSS 全体を変更することなく対象ソフトウェアに再利用

(2)部分的再利用(P) : OSS の一部のみを対象ソフトウェアで再利用

(3)構造変更再利用(SC) : OSS を構造的に変更して対象ソフトウェアで再利用. コードの合成など, 元のファイルやディレクトリの名前や場所が変更されるものを指す.

(4)コード変更による再利用(CC) : ソースコードを変更して OSS を再利用

検出された再利用パターンの分布と平均修正度



- 95%の再利用が修正
- OSSの48%が再利用されていた

Fig. 6: Depiction of detected reuse patterns and averaged modification degree obtained from our experiment. Partial reuse appeared the most, and we found that developers reused only half of the OSS code base with 5% code changes on average.

測定精度

TABLE III: Accuracy of CENTRIS component identification results.

(For 14,997 cases)	Validation result				
	#TP	#FP	#FN	Precision	Recall
<i>Automated verification results</i>					
Paths (V_P)	3,685	N/A	N/A	N/A	N/A
Header files (V_H)	3,286	N/A	N/A	N/A	N/A
Metadata files (V_M)	4,175	N/A	N/A	N/A	N/A
<i>Combined all automated verification methods ($V_P \cup V_H \cup V_M$)</i>					
	8,066	N/A	N/A	N/A	N/A
<i>Manual verification</i>	5,510	1,421	861	0.80	0.86
Total	13,576	1,421	861	0.91	0.94

- OSS コンポーネントのほとんどは変更されて再利用されていたが, CENTRIS は 91%の Precision と 94%の Recall を達成
- ほとんどのケースで再利用されたコンポーネントを正確に特定

今後の課題

- CENTRIS のコードセグメンテーションは, データセットに含まれる OSS の数に影響される

→コンポーネント DB に含まれる OSS の最適な数を見つける

まとめ

- 管理されていない**OSS**コンポーネントは重大なセキュリティ問題や法的問題を増加させる
→**OSS**の再利用を特定することは、ソフトウェア開発実務における課題
- **CENTRIS**は再利用された**OSS**コンポーネントが大きく変更されたり、入れ子になっていても、正確に識別することができた
- **CENTRIS**によって、その脅威を軽減、ソフトウェアの保守性を向上させ、より安全な開発環境を実現することができる

所感

- 思っている以上にコードが修正されて再利用されていることが多い。(自分の中での再利用は, 主にコピペのイメージが強かった)
- 関数に注目して再利用を検出することは, 自分の研究の今後の流れと似ていると感じた