変更追跡機能を用いた静的検査ツールの効果的な利用法

An Effective Usage of Static Check Tools using Change Tracking

渥美 紀寿[*] **桑原 寛明**[†]

あらまし コーディングにおける誤りを早期に発見するため，様々な静的検査ツールが提案され，ツールとして実現されている．しかし，ツールが出力する警告箇所は非常に多く，false-positive の割合が大きいことなどが理由で開発者はあまり利用していないのが現状である．本研究では，ソースコードの変更履歴を追跡し，過去に問題ないと判断された警告箇所を除外するための手法を提案する．OSS を対象に調査したところ提案手法によって確認すべき警告箇所を大幅に削減可能であることを確認した．

1 はじめに

静的検査ツールは対象ソフトウェアを実行することなく解析することが可能でき，不具合の可能性のあるコードを開発工程の早い段階で検査可能である．そのため，静的検査ツールを利用し，定期的にソースコードを検査することは，不具合を早期の段階で検出することができ，ソースコードの品質を向上させるために有用である．

しかし，静的検査ツールによる検査結果は非常に多くの警告を出すため，それらすべてを確認することは非常に労力がかかる．また，false-positive が多いことや思うようにカスタマイズできないなどの理由により，開発者はこれらのツールを使わないことが多い [1]．

本研究では，静的検査ツールが出力する警告箇所を過去のバージョンにおいて確認済みの箇所を取り除くことによって，新たに確認すべき箇所を減らす手法を提案する．これにより，開発者が確認しなければならない警告箇所を大幅に削減することができ，効果的に静的検査ツールを活用することが可能となる．

2 静的検査ツール

C で記述されたオープンソースソフトウェアの openssl と dovecot を対象に静的検査ツール Splint を用いて検出された警告箇所の数と総行数がバージョンの変化に伴ってどのように変化するかを調査した．その結果を図 2 に示した．いずれも行数の変化に伴なって警告箇所の数が変化しており，Splint の警告箇所を修正していないと考えられる．これまで発見された不具合と Splint の警告箇所は関係がなかったと考えられる．しかし，iOS 7.0.6 で修正された不具合 [2] のように静的検査やコンパイラの警告によって検出される単純な問題も発生し得るため，静的検査ツールを用いてチェックすることは重要である．

openssl では 39,140 箇所，dovecot では 42,332 箇所もの警告箇所があり，これらすべてをリリースのたびに確認することは非常に労力がかかる．たとえばヘッダファイル中のプロトタイプ宣言の変更により，その関数を利用している箇所に警告が出る場合があり，変更された部分だけ確認するのでは不十分である．既存の多くのツールでは検査するルールを指定可能であり，これを利用して検査すべきルールを限定することによって確認すべき箇所を減らすことが可能である．しかし，一般に除外しても良いルールが何かを決めることは非常に困難であり，検査すべきルールを選別することは難しい．また，ある箇所では無視して良いが，他の箇所では無

[*]Noritoshi Atsumi, 名古屋大学

[†]Hiroaki Kuwabara, 立命館大学

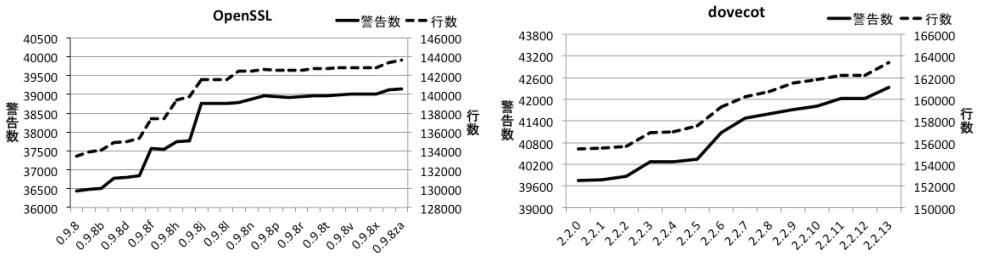

図 1　Splint の警告数の変化

視できない場合もあり，そのような検査すべき対象の細かい選択ができない．

そのため，効果的に静的検査ツールを利用するためには，過去のバージョンにおいて，問題ないと判断された検出箇所を現バージョンにおいても除外でき，修正された場合には再度確認できる仕組みが必要である．

3　静的検査ツールの効果的な利用法

過去のバージョンで問題ないと判断した箇所を現バージョンで除外できるようにするためには，除外したい箇所をコメント等でマークしたり，バージョン間のソースコードを追跡する必要がある．除外したい箇所にコメントを挿入する方法では，その箇所を修正した際に別の問題が発生する可能性があり，除外して良いかどうかを修正時に再度判断する必要がある．しかし，コメントが消し忘れると未確認の問題が隠蔽されてしまう．バージョン間のソースコードの行単位での対応関係を利用する方法では，現バージョンの警告行に対応する前バージョンの行を取得することが可能である．現バージョンで出された警告内容が，前バージョンの該当行の警告内容と同じであれば，それは確認済みとみなすことが可能である．

OSS の OpenSSL と dovecot の各バージョンにおいて新規に検出された警告箇所の数を調査したところ，いずれも全検出箇所の 40 分の 1 以下であり，確認すべき箇所を大幅に削減することができた．ただし，この調査では現バージョンの警告箇所の行に対応する前バージョンの行に警告があれば，対応する警告があったと判定するため，同じ警告かどうかは確認できていない．そのため，警告内容の対応を取らないと正確ではないが，いくつかのバージョンにおいて確認したところ，対応する行に異なる警告が出ることはそれ程多くなかった．

4　まとめと今後の課題

本研究では，静的検査ツールが出力する警告箇所を減らすために，過去に確認した箇所を除外するための方法を提案した．これにより各リリースのタイミングで確認すべき箇所を十分に減らすことが可能となることを OSS を対象とした調査によって確認できた．今後はこれらを自動化するためのツールを実装し，複数の静的検査ツールを効果的に利用できる枠組みを実現する．

謝辞　本研究の一部は科研費 基盤研究 (B) 24300006，若手研究 (B) 24700036 の助成を受けた．

参考文献

[1] B. Johnson, Y. Song, E. Murphy-Hill and R. Bowdidge: Why Don't Software Developers Use Static Analysis Tools to Find Bugs?, In *Proc. of ICSE 2013*, pp.672–681, 2013.

[2] M. Bland: Finding More Than One Worm in the Apple, *ACM Queue*, Vol.12, No.5, p.10, 2014.