C-024

組込機器に搭載する OSS の保守作業の効率向上
Improving the maintenance efficiency for OSS installed in embedded devices

内田 修平† 深井 悠介† 玉田 竜一†
Shuhei Uchida Yusuke Fukai Ryuichi Tamada

1. はじめに

組込機器のソフトウェアは，高機能化に伴って規模が拡大しており，開発工数を削減するために Open Source Software（OSS）の活用が進展している．一方で，製品化後の保守，特に，製品化後に発見される OSS の脆弱性や不具合を修正したバージョンへの更新作業の効率化が課題となっている．そこで，カバレッジ情報を使って修正内容が機器の動作に与える影響を明らかにすることにより更新作業を効率化することを目指している．

今回，修正コードの差分と機器の動作を示すコードカバレッジを比較することにより修正内容が機器の動作に与える影響を明確にする環境を構築したので，本稿で報告する．

2. 富士電機における取組

当社では，クラウド等に接続する通信端末をはじめとする製品に対し，OSS を適用している．適用にあたり，OSS を効率よく試験および保守するための「自動試験環境」の構築を進めている．本環境は以下で構成している．

(a) OSS を搭載する組込機器にて自動で試験プログラムを実行し，その結果を取得する仕組
(b) これらを開発プロセスにおける適切なタイミングで実行する仕組

仕組(a)では，試験結果の取得だけではなく，試験の網羅性を確認するため，対象機器において実行されるソースコードのカバレッジを計測している．この自動試験環境により，製品化後のソフトウェアの更新にかかる工数の削減を目指している．

3. 取組にあたっての課題

組込機器では，顧客の損失につながる機器の非稼働時間はなるべく最小にしたいため，更新回数を抑えたいという要望がある．このため，OSS の修正内容が組込機器に与える影響を把握し，影響を与える修正を特定することで，更新回数を抑えている．しかし，この確認作業は，OSS コミュニティの Web ページ，リポジトリ等を定期的にチェックし，修正情報を入手して，その内容が組込機器に影響を与えるかを確認する必要がある．この作業は，膨大なコード量の中から組込機器が実行する箇所を特定し，OSS 修正コードとの関係を明らかにしなければならず，これにかかる工数（1 人月程度）が大きく，更に更新頻度が多い場合には大きな負担となっていた．また，この作業を行うには当該 OSS の知識が必要で，その人材を適宜割当てなければならなかった．今回，この確認作業の効率化を課題としている．

4. 解決策

組込機器に影響がでる場合は，実行するソースコードに変更が加えられる場合である．そこで，実行するソースコードと修正するコードを比較できれば，修正内容による組込機器の影響の有無の判定ができると考えた．その手順を次に示す．

(1) 膨大な OSS のコードの中から組込機器が実行する箇所を特定するため，試験時のカバレッジ情報を使用する．
(2) OSS の修正の特定は，修正パッチファイルから変更箇所を特定する．
(3) OSS の修正が，組込機器に影響を与えるかどうかの判定は，(1)で特定した組込機器の実行箇所と(2)で特定した OSS の変更箇所を比較し，重複する部分がある場合に影響があるものと判断する．

この方法によると，作業の自動化が可能であり，前節で示した課題を解決できる．

5. 実施内容と結果

5.1 対象

今回，対象とする OSS は，当社機器上で動作する OSS の中で膨大なソースコード量があり，更新頻度が高い Linux Kernel（以下，カーネル）とした．また，解決策の内容が実現可能であるかを把握するため，表 1 に示すカーネルと，その主要機能を試験するプログラムを用意した．

表 1 取組対象

項目		対象
更新対象 OSS	名称	Linux Kernel
	更新前のバージョン	3.18
	更新後のバージョン	4.14
対象機器上で実行するソフトウェア		カーネルの主要機能の試験プログラム

5.2 実施方法

はじめに，対象機器が実行するソースコードを確認するため，GNU Compiler Collection（GCC）の標準ユーティリティとして付属する Gcov [1]を有効にしてビルドしたバイナリに対して，試験プログラムを実行してカバレッジを計測した（図 1）．これについては，既報[2]に記載している．

次に，更新前のバージョン（3.18）から更新後のバージョン（4.14）に更新された箇所を特定するため，Unified Format [3]に従ったパッチファイルを作成した．Unified Format の文法を図 2 に示す．記載される主な内容は，変更ファイル，変更箇所を示す行，削除箇所，追加箇所である．

そして，この変更箇所の情報を使い，カバレッジ計測結果（図 3 の(i)）の実行済箇所とパッチファイル（同図(ii)）の変更箇所が重複する部分を抽出することで，パッチにより変更される箇所が組込機器によって実行されると判断し，その結果を出力した（同図(iii)）．これを Linux kernel の各ファイルに対して自動実行を可能にし，一覧を生成した．

† 富士電機株式会社 Fuji Electric Co., Ltd.

この重複部分の抽出は，次の手順で実施した．

(1) カバレッジ計測結果の各ファイルに対して，実行済範囲の開始行と終了行をリスト構造で抽出
(2) 同様にパッチの各ファイルに対して，変更範囲の開始行と終了行をリスト構造で抽出
(3) 抽出した実行済範囲と変更範囲が以下の条件となる時を影響があると判定
  ・変更範囲の前半と実行範囲の後半が重複する
  ・変更範囲が実行範囲に含まれる
  ・変更範囲の後半と実行範囲の前半が重複する
  ・変更範囲が実行範囲を含む

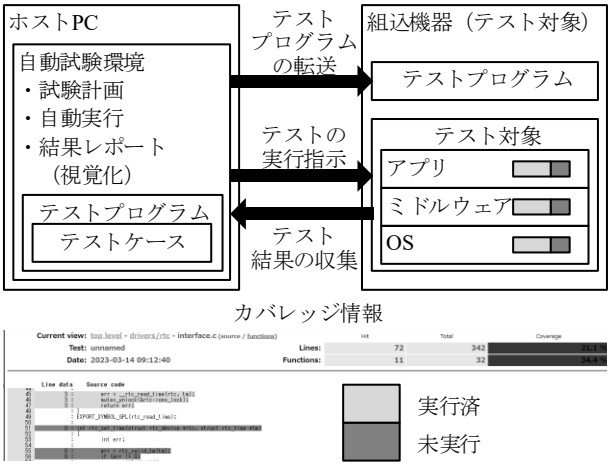


図 1 カバレッジの計測

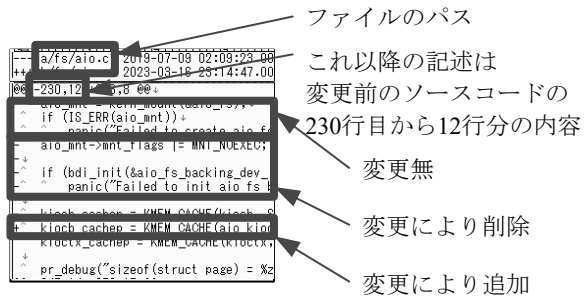
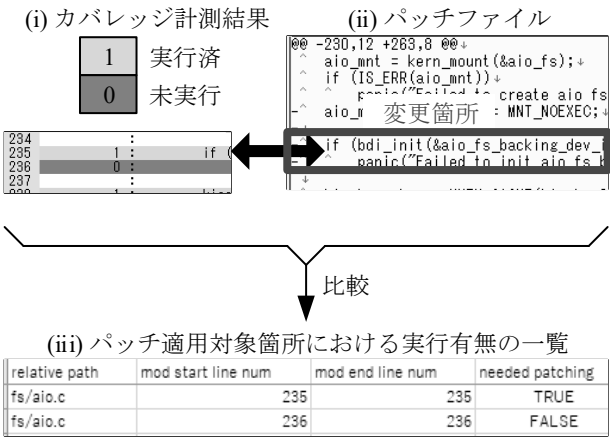


図 2 Unified Format



relative path	mod start line num	mod end line num	needed patching
fs/aio.c	235	235	TRUE
fs/aio.c	236	236	FALSE

図 3 実行済箇所と変更箇所の比較

5.3 結果

5.1 節に示した対象について，パッチによる変更箇所が組込機器によって実行されるかどうか，表 2 に示す内容で出力でき，実際の実行結果と同じになること確認した．

また，パッチによる修正対象かつ実行済の箇所を有するソースコードファイルは，カーネルのソースコード全 2,325 点中 1,436 点であった．パッチ修正対象有無と機器での実行有無との各組合せに該当するソースコードのファイル数を表 3 に示す．

表 2 実行有無の一覧の内容

相対パス	開始行	終了行	実行済
（略）			
fs/aio.c	235	235	はい
fs/aio.c	236	236	いいえ
（略）			

表 3 修正対象と実行有無との各組合せに該当するソースコードのファイル数

		機器での実行有無		
		実行	不実行	計
パッチによる修正対象	対象	1,436	150	1,586
	対象外	416	323	739
	計	1,852	473	2,325

6. 考察

今回，パッチによるカーネルの更新が対象機器の動作に与える影響範囲が，ソースコードの割合で約 6 割（全 2,325 ファイル中 1,436 ファイル）であることを判断するのに数分とかからなかった．また，この仕組により，OSS の知識が無くても，OSS の更新が対象機器の動作に与える影響を数値で示すことが可能になり，作業の効率化に貢献できることがわかった．

一方で，Linux Kernel バージョン 3.18 から 4.14 までの間では対象機器が利用する多くの機能で修正が行われていることから，チップベンダーが提供する BSP（Board Support Package）に含まれる Linux を利用する場合には，変更箇所が多く含まれ，更新回数削減は難しいことがわかった．

7. おわりに

今後は，比較するバージョンが近く，特定の機能しか実行していない OSS の場合に更新回数が削減できるケースがあるか試す．また，テストケースとカバレッジ情報を用いて，変更箇所の動作確認に必要な試験項目を特定し，試験の自動化ができていない手作業の試験の効率化に向けて取り組む．

参考文献

[1] Gcov, https://gcc.gnu.org/onlinedocs/gcc/Gcov.html
[2] 内田 修平, 深井 悠介, 玉田 竜一, “組込 Linux 搭載機器におけるカバレッジ測定方法”, 情報処理学会第 86 回全国大会, 2024.
[3] Unified Format (Comparing and Merging Files), Free Software Foundation, https://www.gnu.org/software/diffutils/manual/html_node/Unified-Format.html