G-5

組込みシステムを考慮したコードレビュー支援手法

A method of support the code review considering the embedded system

○増田智樹[1]，平山雅之[2]

Tomoki Masuda[1],Masayuki Hirayama[2]

abstract: Code review is important in software development. However, a expression form or language is different from specification and source code. Moreover, there is a limitation to the source code, such as memory size in embedded system. This report proposes a code review supporting method using function information of both specification and source code.

1.　まえがき

ソフトウェア開発において，仕様書とソースコードを照合して要求漏れや実装漏れ等を見つけるコードレビューは品質保持面で必須の作業である．しかし仕様書-ソースコード間の対応を逐一確認する作業は多くの工数を必要とする．この原因としては仕様書とソースコードでそれぞれの記述表現様式や表現粒度の違いがあり，そのために対応に多大な時間がかかることが考えられる．また，組込みシステムにおいてはメモリ制限が存在するため，ソースコードを簡潔に書くことが求められ，結果として機能説明が省略されておりコード上の機能読解が困難であることも原因の 1 つであると考える．

仕様書-コード間のレビューについては様々な支援方法が利用されているが，これらの多くはコードから仕様情報を逆生成し確認する方法が中心となっている．この方法では本来の仕様書に記述されている機能情報と逆生成された機能情報との情報粒度が異なり意味解釈が一意に定まらないという問題が生じてしまう．そこで本研究では「仕様書に記述された機能情報」と「ソースコード上で表記された機能情報」の両方の機能情報を関数名情報で橋渡しする．これにより双方の情報粒度や表現のギャップを埋め，仕様書-ソースコードの機能対応付けを効率的に支援する手法を提案する．

2.　提案手法

2.1 概要

本提案手法はコードレビュー時に仕様書に記述された機能情報とコード上に表記される機能情報の間に情報粒度の差異が小さくなるように，それぞれに記載されている関数名を基に抽出・対応付けを行なうものである．特に組込みシステムではサイズ制限からコード上でコメント文による機能説明が省略されている場合が考えられるため，ソースコード側ではコードとは別に補足情報を記載したリストを用意する．それぞれから抽出した機能情報を仕様書-ソースコード機能対応リストの形に整理する．機能対応リストには仕様書，ソースコード両方に記述されている全ての機能が対応付けされた状態で記述されている．これを用いて仕様書-コード間の対応付けや機能読解を行う事で時間的な短縮が実現できる．また，対応付けがされなかった機能は要求漏れや実装漏れとして発見も可能である．

提案手法の概要図を以下の図 1 に示す．

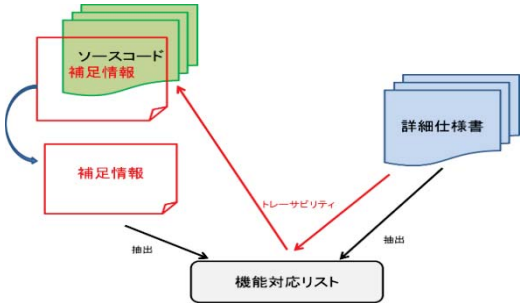


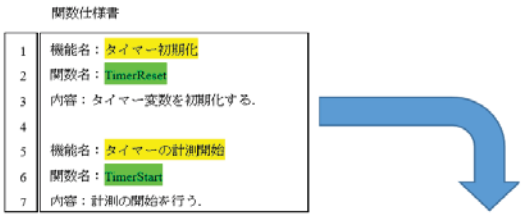
Figure.1 Overview of the proposed method

2.2 仕様書側抽出方法

本手法では，抽出する仕様書は詳細仕様書レベルのものであり，Word で記述されたものを対象とする．その上で，関数名と関数の機能説明部分に対して抽出の標的となる印，word 機能の蛍光ペンを抽出印とし関数名に付ける．関数名に蛍光ペンの付いた word 仕様書に対して word VBA で作成したマクロにより機能情報データの抽出を行なう．抽出をする項目は関数名，機能説明，位置情報とする．

抽出印は仕様書作成段階で抽出する．抽出例を以下の図 2 に示す．

1：日大理工・院(前)・情報　2：日大理工・教員・情報



関数仕様書側関数情報

	仕様書					
ID	頁	行数	機能名	複雑度	型	関数名
1	1	1	タイマー初期化			TimerReset
2	1	5	タイマーの計測開始			TimerStart

Figure2 Specification side extraction example

2.3 ソースコード側抽出方法

ソースコード側では組込みシステムを考慮してコメント文が記載されていないソースコードを対象とする.補足情報の例を図 3 に示す.

ID	8	関数名	start_pwm
位置情報	pwm.c		9～40
機能概要	PWM設定と出力開始		
仕様ヘッダ	pwm.h		
デバイス	モータ		
呼出関数	set_duty		
優先度	高	pwm出力開始関数のため	
機能説明	L:14～17	デューティ比の最大値100	
	L18～21	デューティ比の最小値0	
	L25	isfrontFlg = 1の時に前進	
	L29	isfrontFlg != 1の時に後進	
	L35	set_dutyを介してduty値取得	
	L36	PWM出力開始.モータ回転	

Figure3 Supplementary information example

補足情報にはソースコードから関数名，機能が含まれているフォルダ名と位置情報，関数内で呼び出している内部関数，仕様ヘッダ，機能概要，重要な処理文の説明，優先度，外部デバイスなどの項目を記載する.

補足情報は開発者がソースコードを記述した際に合わせて関数名，位置情報，内部関数を自動抽出する．その他の項目は自動抽出後にコードを参照しながら開発者が埋めていく.

2.4 機能対応方法

仕様書，ソースコード両方から抽出した機能情報の対応付けはそれぞれの関数名を基に結合を行ない，機能関係リストを作成する.

作成した機能関係リストには仕様書，コードのそれぞれから抽出し結合した機能情報が記載される．このため，この機能関係リストを見れば要求されている機能が仕様書のどこに記述されているか，それに対応している機能がどこで実装されているかを一目で確認することができる．また，コード側では関数の内部関数を記載しているため，その関数の凡その処理や他関数との関連が把握可能であり，内容確認にも利用することが出来る.

機能関係リストの例を以下の図に示す.

仕様書				ソースコード				
ID	頁	機能概要	優先度	型	関数名	行数	ファイル名	内部関数ID
1	7	PWM設定と出力	高	int	start_pwm	30	pwm.c	2
2	7	Duty値設定	中	void	set_duty	21	pwm.c	
3	5	SWの状態取得	高	int	sw	30	main.c	4

Figure4 Examples of function-related list

3. 机上実験

今回提案した手法により，レビューにかかる負担がどの程度削減が行えるかを検証するために机上実験を行なった．実験対象は学生が作成したクローン検出プログラム(仕様書 17 頁，ソースコード 2060 文字，C++記述，26 機能)とする．これに対し，仕様書に記述されている機能と同じ関数名で実装されている機能をコード中より探し，内部関数の抽出にかかる時間を機能関係リストの有無に分けて測定した.

その結果，機能対応付けは両方とも全ての機能を対応付けすることが出来た.

次に抽出時間では，機能関係リストを用いなかった場合では実験に要した時間は約 38 分であった．　次に機能関係リストを用いた場合では実験に要した時間は約 16 分であった.

実験結果から，リストを用いることで関数の探索と機能理解にかかる時間の短縮が行われることで機能対応にかかる時間を削減出来たと考えられる.

4. まとめ

本研究ではレビュー時に多大な時間を必要とする機能対応の時間削減を行なうために仕様書に記述されている機能情報とコード上に記述されている機能情報を直接利用した機能関数リストを提案した.

簡易机上実験により，機能関係リストは関数探索と機能理解にかかる時間の短縮に効果があることが分かった.

今後は抽出，機能対応を行なうツールの実装を行ない，レビュー全体の時間短縮を考慮した評価実験を行なっていく.

5. 参考文献

[1]田原貴光，林　晋平，他　編，″仕様書と Java ソースコードの構造の類似性に基づく対応付け″,情報処理学会研究報告，2008(29),139-146,2008-03-17