2L-07

# 機能情報を利用した仕様書-ソースコード間の機能対応方法の検討

増田　智樹[†]　　伊賀　達人[‡]　　平山　雅之[†‡]

[†]日本大学大学院　理工学研究科　[‡]日本大学　理工学部

## 1. はじめに

仕様書とソースコードを照合して行なうコードレビューは，要求漏れや実装漏れを見つける等，ソフトウェアの品質保持面で必須の作業である．しかし，こうしたレビューでは仕様書-ソースコード間の対応を逐一確認するために多くの工数を必要とする．特に，仕様書，ソースコードそれぞれの記述表現様式や表現粒度の違いがこの原因の一つになっていると考えられる．

仕様書-ソースコード間のレビューについては様々な支援方法が提案・利用されている．これら従来法の多くはコードから仕様情報を逆生成し，確認する方式が中心となっている．しかし，この方法では仕様書に記述された機能情報とコードから逆生成された機能情報の対応付けを行う際に，情報粒度が異なり意味解釈が一意に定まらないといった問題が生ずる．この結果，要求や実装面での漏れや追加の有無の判断が難しくなる場合がある．これに対し，文章の類似性を用いて対応付けする方法が提案されている[1]．しかしこの方法では辞書機能や単語分解などの処理が入るため，誤った意味解釈がなされる可能性がある．このため，本研究では「詳細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し，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検討している．

## 2. 提案手法の概要

提案手法は以下の 3 つの手順から構成される．

手順①：詳細仕様書からの機能情報抽出．

手順②：実装コードからの機能情報抽出．

手順③：関数情報を用いた情報①②の連結

図 1 に全体の流れを示す．詳細仕様書と実装コードからそれぞれの機能関連情報の抽出を行ない，情報が一致したものを対応付けする．対応付けされたものの双方の機能の関係が一目で分かる機能関数リストを設ける．これにより機能の対応付けをより確実に行えるようにする．

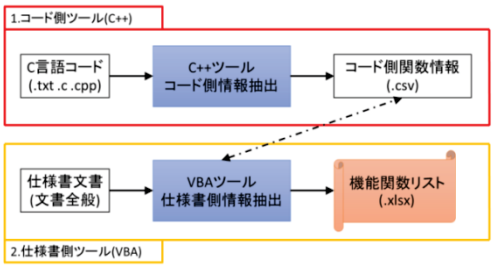



図 1 提案手法の概要図

## 3. 機能情報抽出方法

### 3.1　仕様書からの情報抽出

提案法の手順①では，自然言語で記述された仕様書から機能情報を抽出する．ここでは対象とする詳細仕様書は word で記述され，かつ関数名が予め記述されていることを前提とする．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，本稿では word 仕様書が対象のため，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．図 1 に例を示す．①③が機能説明，②④が関数名である．

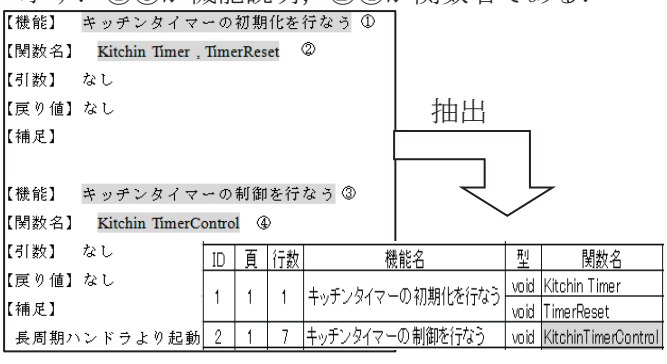


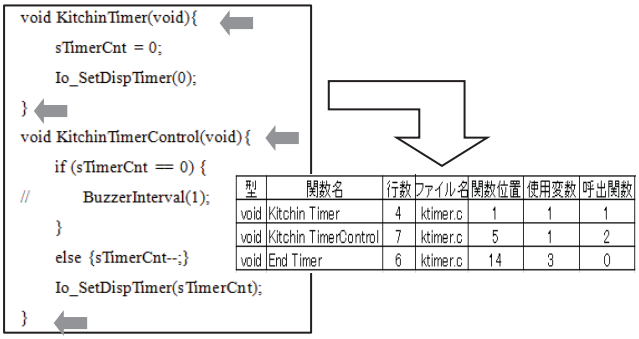

図 2 仕様書の機能情報抽出例

### 3.2　ソースコードからの機能情報抽出

提案法の手順②では実装したソースコードから機能情報の抽出を行なう．本手法ではコードは構造化分析設計に基づき C 言語で記述されたもの対象としており，抽出範囲は詳細仕様書との対応を考慮し，関数単位での抽出を行なう．抽出方法は関数本体の範囲を示す中括弧({,})を

A method of function mapping between specifications and source code using functional information
Tomoki Masuda†, Tatsuhito Iga ‡, Masayuki Hirayama†‡
†Graduate School of Nihon Univ.
‡,Nihon University.

抽出印としてソースコードを関数ごとに分割を行ない，関数ごとに情報の抽出を行なう（図 2）．




| 型 | 関数名 | 行数 | ファイル名 | 関数位置 | 使用変数 | 呼出関数 |
|---|---|---|---|---|---|---|
| void | Kitchin Timer | 4 | ktimer.c | 1 | 1 | 1 |
| void | Kitchin TimerControl | 7 | ktimer.c | 5 | 1 | 2 |
| void | End Timer | 6 | ktimer.c | 14 | 3 | 0 |

図 3 コードの機能情報抽出

## 4. 機能対応方法

前章に示す方法で仕様書とソースコードから抽出した機能情報の中から関数名を基に機能情報の結合を行ない，機能関数リストを作成する．

例えば図 2，3 の例では Kitchin Timer という関数名をキーに双方の情報を連結している．このように詳細仕様書で要求された 1 つの機能に対し，1 つの機能で実装されるのが理想である．しかし実際には 1 つの要求機能を複数の関数に分けての実装や，逆に複数の要求機能を 1 つの関数にまとめて実装する場合がある．そこで各場合を考慮した対応付けを図 4 に示す．

| 仕様書 | | | | ソースコード | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ID | 頁 | 行数 | 機能名 | 型 | 関数名 | 行数 | ファイル名 | 関数位置 | 使用変数 | 呼出関数 |
| 1 | 1 | 1 | キッチンタイマーの初期化を行なう | void | Kitchin Timer | 4 | ktimer.c | 1 | 1 | 1 |
| | | | | void | TimerReset | 3 | ktimer.c | 22 | 0 | 0 |
| 2 | 1 | 7 | キッチンタイマーの制御を行なう | void | KitchinTimerControl | 7 | ktimer.c | 5 | 1 | 2 |
| 3 | 1 | 7 | キッチンタイマーをstartする | void | KitchinTimerControl | 7 | ktimer.c | 5 | 1 | 2 |
| 4 | 1 | 5 | キッチンタイマーをstartするendする | | | | | | | |
| 5 | | | | void | End Timer | 6 | ktimer.c | 14 | 3 | 0 |
| | | | | ⋮ | | | | | | |

図 4 機能対応リストの例

図 2-①の 1 つの要求機能に対し，図 2-②，Kitchin Timer，TimerReset ように実装関数が複数ある場合，図 4 の ID-1 のように関数名行を挿入することで複数の実装機能を対応付ける．これにより要求機能に対する複数の実装関数が簡便に確認できる．また，図 4 の ID-2，3 のように実装関数が他の要求機能と重なっている場合，重複を示すために色付けを行なうことで重複していることを表し，重複先を示す．これにより重複している機能を容易に把握できる．

## 5. 机上実験

上記の提案法を用いて実際の仕様書，コードでも対応付けが可能であるか検証するために机上実験を行なった．実験は簡易な電子温度計測システムの仕様とコードを対象とした．

### 5.1 実験方法

手順①では，仕様書内の【関数名】と書いてある行を機能リストの関数名欄に，【機能】と書いてある行を機能名欄に抽出した．手順②では C 言語により記述された 3 つのプログラムから関数毎に機能情報抽出を行なった（図 5）．

| ID | 頁 | 行数 | 機能名 | 型 | 関数名 | 行数 | ファイル名 | 位置 |
|---|---|---|---|---|---|---|---|---|
| 1 | 30 | 25 | 電子温度計の仕 | | thermometer | | | |
| 2 | 31 | 22 | 周辺デバイスの | void | intialize | 23 | thermometer_main.c | 193 |
| 3 | 32 | 10 | ADC変換の割り | void | fin_adc | 4 | thermometer_main.c | 138 |
| 4 | 33 | 8 | ADC AN1に変換 | void | start_adc | 4 | manage_temp.c | 30 |
| 5 | 34 | 8 | ADC AN1の変換 | 16 | read_adc | 8 | manage_temp.c | 43 |
| 6 | 35 | 10 | SW2押下げ割り | | notice_sw | | | |
| 7 | 36 | 11 | タイマ1完了割 | void | fin_timer1 | 4 | thermometer_main.c | 167 |
| 8 | 37 | 11 | タイマ2完了割 | void | fin_timer2 | 4 | thermometer_main.c | 181 |
| 9 | 38 | 11 | ADC変換結果を | void | get_temp | 26 | manage_temp.c | 60 |
| 10 | 39 | 11 | 最高最低温度を | void | renew_temp | 10 | manage_temp.c | 95 |
| 11 | 40 | 9 | 最高最低温度を | void | reset_temp | 5 | manage_temp.c | 114 |
| 12 | 41 | 11 | disp_modeに応 | void | disp_temp | 20 | manage_temp.c | 137 |
| 13 | 42 | 8 | 液晶ディスプレ | void | IntialiseDisplay | 33 | lcd.c | 48 |
| 14 | | | | void | main | 70 | thermometer_main.c | 60 |
| 15 | | | | void | notice_sw2 | 4 | thermometer_main.c | 152 |
| 16 | | | | void | DisplayString | 33 | lcd.c | 94 |
| 17 | | | | void | LCD_write | 7 | lcd.c | 137 |
| 18 | | | | void | LCD_nibble_write | 37 | lcd.c | 154 |
| 19 | | | | void | DisplayDelay | 8 | lcd.c | 197 |

図 5 机上実験結果

### 5.2 実験結果及び考察

机上実験の結果，仕様書から 13 個，ソースコードから 17 個の機能をそれぞれ抽出した．そのうち，仕様書からは ID-1「電子温度計の…」と ID-6「SW2 押下げ…」が抽出されているが，そこで記された関数名に該当する関数がコードからは抽出されずに実装漏れであると判定された．また ID-14～19 はコード上では実装されていたが，該当する関数名が仕様書側に無いため要求漏れであることがわかった．仕様書，ソースコードの記載内容を精査した結果，ID-1 は ID-14「main」関数と，ID-6 は ID-15「notice_sw2」と対応していることがわかった．実装漏れの原因は実装時に関数名が変更されたためであった．また，ID-16～19 に関する要求漏れは ID-13 のディスプレイに関して機能追加が発生した結果，対応する要求機能が無かったことがわかった．

以上のように，機能関係リストを用いることで，機能対応と仕様書-コードの機能漏れを抽出できることを確認した．

## 6. まとめ

本稿では仕様書，コードの記載情報を利用した機能関係リストにより機能対応付けを行なう手法を紹介した．また机上実験より，この手法を利用し，実際の仕様書-コードを対象に機能の対応付けが可能になることを確認した．今後は抽出した情報を基に，レビュー時に利用できるレビュー支援情報の提示を検討していきたい．

**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