Feature Locationを活用したソフトウェア機能の文書化に向けて

風戸広史[†1] 大島剛志[†1]

リエンジニアリングやアジャイル開発の過程ではしばしば，ソースコードに実装済みの機能を文書化する必要が生じる．Feature Location (FL) は機能とソースコード上の要素の対応関係を識別するためのソフトウェア保守技術であるが，このような文書化への応用に関する議論は少ない．そこで，本稿では文書化への応用に向けたFL技術の要件とその評価方法を考察する．

Towards Locating and Documenting Features in Source Code

Hiroshi Kazato[†1] and Tsuyoshi Oshima [†1]

In the course of reengineering or agile development, developers often have to recollect and document the features that are already implemented in source code. Feature location (FL) seems a promising technique to achieve such documentation, but there are few discussion on the usage. In this paper we explore some requirements and evaluation criteria for FL techniques for the use of documenting software features.

1. はじめに

ソフトウェアの開発過程ではしばしば，開発者やステークホルダの間のコミュニケーションためにソースコードに実装済みの機能を文書化する必要が生じる．たとえば，既存の情報システムを刷新するリエンジニアリングでは，新しいシステムを企画するために既存のシステムの機能に関する文書を参照したい．しかし，そのような文書が残っていない，あるいはリリース後の変更が文書に反映されていない場合がある．また，アジャイル開発では文書よりも動作するソフトウェアを重視するが，顧客の要望によっては文書も作成することを謳っており，リリースの前に実装内容を文書化する状況が起こり得る．

実装済みのソフトウェアに対して文書を作成する場合には，開発者が別の作業に移ったために機能や実装の詳細を忘れていたり，プロジェクトや組織をすでに去っていたりすることが問題となる．失われた機能とソースコードの対応関係を別の開発者が理解するためのコストは高価であり，その支援が求められる．

このような文書化の支援のために，我々はFeature Location (FL) 技術が応用できると期待する．FLはソフトウェアに実装された機能要求 (Feature) と対応するソースコード上の要素を特定する開発者の作業であり，ソフトウェア変更プロセスおける作業の一つと考えられている[1)]．このため，機能の包括的な理解や文書化の文脈からFLを捉え，評価する試みは少ない．本稿では文書化に関するFL技術の応用について我々の意見を述べ，その評価について考察する．

2. 変更支援のための FL 技術の評価

ある変更要求が与えられたとき，FLの目的はソースコード上で変更が必要となる中心的な要素を見つけることであり，その要素から変更影響分析 (Impact Analysis; IA) を行い変更が必要となる要素の範囲を見積もる．この文脈では，FL技術が修正すべき要素のうち少なくとも1つを誤りなく見つけ，IA技術がその要素の周辺から修正すべき要素を過不足なく見つけることが効果的な支援となる．

Ditらは FL技術に関する広範なサーベイの結果をもとに，FL技術の比較評価が難しい状況を指摘し，FLのベンチマークを提案した[2)]．このベンチマークは変更要求に対するFLの結果が修正範囲の見積もりに利用できるかどうかを評価するために，オープンソースソフトウェアの実際の変更過程を題材とし，過去の機能改善要望から変更要求を，改版履歴から実際の変更箇所をそれぞれ抽出したものである．この変更箇所を正解集合とし，情報検索技術の評価指標である適合率 (precision)，再現率 (recall) を流用してFL/IA手法の比較評価を行うことをDitらは提案している．

†1 日本電信電話株式会社 ソフトウェアイノベーションセンタ
NTT Software Innovation Center

3. FL 技術を応用した機能の文書化

機能を文書化する場合は変更要求により修正される箇所ではなく，その機能に関する正常系や条件分岐，例外処理を構成する要素をすべて特定したい．

我々はこのような文書化を支援するため，動的解析によって機能の実装箇所を包括的に特定する FL 技術を提案した[3),4)]．3) では多層システムにおける機能の実装箇所を特定するため，複数の機能の実行を含んだシナリオを用意し，シナリオとその実行時に動作する各レイヤの要素の関係を観測する．この関係に形式概念分析を適用すると，特定の機能に固有の要素や複数の機能が共有する要素を階層的に抽出できる．4) では特定した要素群の静的構造や振る舞いをビューとして可視化し，文書化を支援する．動的解析に基づく FL では変更箇所に限らず，機能に関連する多くの要素を抽出できる特性がある．一方で，実行されない要素は分析結果に現れないため，特定の条件分岐や例外処理に関連する要素がシナリオ内で実行されず，機能との対応関係が特定できない可能性がある．

前述のベンチマークを用いると，我々の FL 手法は変更要求の対象となる機能に関連する要素をすべて特定しようとする．一方，正解集合は変更要求によって修正された要素のみであるため，ほとんどの変更要求で適合率が 1%を下回る．この結果は FL 手法の目的とベンチマークの作成方法が合致しないことに起因しており，機能の理解や文書化に必要な要素群を正解集合とする新たなベンチマークが必要である．

4. ソースコードと機能の対応付けの評価

機能に対応するソースコード上の要素をすべて識別する問題を**プログラム理解型の FL 問題**と呼び，その評価のためのベンチマークを考えたい．現時点では構想段階であるが，そのようなベンチマークの正解集合には以下の性質が必要だと考える．

- **機能 (Feature) の定義**．曖昧さを残さないように機能 [⋆1]を記述し，その単一の記述をもとに各種の FL 手法の入力を生成可能にする．具体的には，情報検索技術のためにクエリとなるキーワードや文を抽出し，静的解析のために探索の開始点となる要素を特定し，動的解析のためにシナリオを作成できることが求められる．
- **機能と要素の関連性**．機能の実装箇所に該当する全ての要素を列挙する．FL 手法の出力や第三者の判断と比較するために，各要素がそこに含まれる根拠，機能に対する貢献点が明らかであることが望ましい．
- **要素の粒度**．機能に対応づけるソースコード上の要素の粒度は，実装や再利用の単位である関数やメソッドの単位が適切と考える．Koschke らは動的解析に基づく FL 技術において関数やメソッドの単位から文や制御構造の単位まで粒度を下げ，FL で得られる情報量の違いを研究した[5)]．関数やメソッドが複数の機能の実装箇所に相当し，凝集性が低い場合はその内部まで粒度を下げることが有効である．その一方で，粒度を下げると目視での確認や動的解析のコストが大きくなる．

単一の企業や大学でベンチマークを開発するとコストを要するだけでなく，正解集合の性質や評価方法に偏りが生じやすく，妥当性への脅威となり得る．このようなベンチマークは技術革新を促進し，産業上の発展を助ける側面もあることから，コミュニティの議論の中で構築し，改善していくことを提案したい．

5. おわりに

本稿では FL 技術を応用した機能の文書化について筆者らの考えを述べ，その評価に必要となるベンチマークの要件を考察した．ワークショップの討論では FL や IA，トレーサビリティ復元等の関連技術がどのような応用を前提としているか，その評価のためにはどのような実験方法とデータが必要かを議論したい．評価方法は手法の目的によって異なるが，プログラム理解型の FL 問題にある程度の共通性があり，ベンチマークの構築で連携できることを期待する．

⋆1 FL における機能とはソフトウェアに実装済みの機能要求であり，外部から入力と応答の系列によって観測可能なものを指す．

参考文献

1) Rajlich, V. and Gosavi, P.: Incremental change in object-oriented programming, *IEEE Software*, Vol.21, No.4 (2004).
2) Dit, B., Revelle, M., Gethers, M. and Poshyvanyk, D.: Feature Location in Source Code: A Taxonomy and Survey, *JSME* (2011).
3) Kazato, H., Hayashi, S., Okada, S., Miyata, S., Hoshino, T. and Saeki, M.: Feature Location for Multi-Layer System Based on Formal Concept Analysis, *Proc. CSMR* (2012).
4) Kazato, H., Hayashi, S., Okada, S., Miyata, S., Hoshino, T. and Saeki, M.: Toward structured location of features, *Proc. ICPC* (2012).
5) Koschke, R. and Quante, J.: On Dynamic Feature Location, *Proc. ASE* (2005).