セッション紹介：ソースコードと機能の対応関係を特定する技術

石尾　隆[†1]　林　晋平[†2]

大規模なソフトウェアに対して適切な変更やテストを実施していくためには，ソフトウェアに対する要求や，開発者が変更したい機能を実現するソースコードを特定する作業が不可欠である．このような作業を支援するために，Concept Location，Feature Location，Impact Analysis，Concern Location，Traceability Recovery などの解析技術が提案されている．本セッションでは，これら技術の開発における問題意識，手法の実装や評価実験における技術的な課題について議論し，効果的な技術開発の方法に関する知識を整理したい．

Session: Identifying Links between Source Code and Software Features

Takashi Ishio[†1] and Shinpei Hayashi[†2]

To maintain a large scale software, developers must identify source code where implements a requirement or a feature to be modified. Program analysis techniques including concept location, feature location, impact analysis, concern location and traceability recovery are proposed to support linking source code with requirements and features. In this session, we would like to discuss technical issues to design, implement and evaluate novel techniques in this area.

1. はじめに

ソフトウェアは，社会状況や企業活動の変化に応じて継続的に変更が行われる．そのため，多くのソフトウェア開発者は，自身が開発したものではないソフトウェアに対しても，その機能の実装方法を理解したうえで適切な変更を施すことを求められる．そこで，ソフトウェアのソースコードと，ソフトウェアの機能の名称や変更要求，設計文書，操作説明書，テストケースのような周辺情報とを自動的に対応付ける技術が注目されている．

ソースコードと周辺情報との対応付けは，一般的に，ソフトウェアに実装された概念（Concept）と，その概念のソースコードや文書上での表現（Extension）とを対応付けることによって行われる[5]．この対応付けを行うため，Concept Location，Feature Location，Impact Analysis，Concern Location，Traceability Recovery といった様々な技術が提案されている．本稿では，まず 2 章でこれら技術間の関係を整理し，3 章で，これらの技術に共通する課題の例を挙げ，セッションでの議論の準備とする．

†1 大阪大学
Osaka University
†2 東京工業大学
Tokyo Institute of Technology

2. 技術間の関係

Rajlich は Concept Location を，開発者がソフトウェアの変更作業を開始したとき，変更の対象になるモジュールを少なくとも 1 つ特定する作業と定義している[5]．ソフトウェアの変更要求は，通常，ソフトウェアに実装された何らかの概念（たとえば「銀行口座の残高」や「利率の計算」,「メッセージの出力」など）を取り扱うことから，Concept Location は変更要求に含まれた概念を見つけ出し，その実装に関与している変数や関数，メソッド，クラスなどを少なくとも 1 つ特定する，変更作業の第一歩となる作業として位置付けられる．

Feature Location は，Concept Location の一種であるが，ユーザが実行するかどうかを選択できるような機能（Feature）の特定に注目したものである[6]．たとえば，ある機能を実行した場合と，実行しなかった場合での動作を比較するなどの方法によって，機能が実装されている可能性が高いソースコード位置を特定

する．ユーザが実行を制御できることを利用していることを除けば Concept Location との違いはなく，論文によっては明確な区別は行わない．

ソフトウェアの変更を適切に行うには，Concept Location や Feature Location によって見つかったモジュールを手がかりとして，現在の機能の実装方法を理解し，現状に合わせたソースコードの変更方法を決定しなくてはならない．ソースコードの変更方法を決めた結果，変更を施さなくてはならないすべての要素を特定する作業を Impact Analysis と呼ぶ[2),5)]．ここでの Impact Analysis は，変更の影響が波及する範囲の特定を意味しており，回帰テストを行うべきコードの範囲を調査する作業とは定義が異なる点に注意が必要である．

Concern Location は，ソフトウェアのある関心事の実装を理解することを目標とした技術である．ソフトウェアの変更を想定しないかわりに，関心事をソフトウェアから取り除くとしたらどこを削除・修正する必要があるかという観点で関心事の特定を行う[1)]．

Traceability Recovery は，互いに関係のあるソースコードや文書を結び付ける技術である．ソースコードや文書の内容を概念の名前などに結び付け，同一概念に関係したソースコードや文書の集合を得る技術とみなすこともできる[5)]．

3. 議　　論

5 つの技術は，その手法や動機は様々であるが，いずれもソースコードと「機能」の名前や設計文書などとの対応関係を求める検索問題の一種である．Concept Location と Feature Location は，探したい機能を表現した文字列などを入力として，目的のモジュールを少なくとも 1 つ発見することを目標としている．Impact Analysis と Concern Location は，変更対象あるいは関心事に属するすべてのモジュールを発見することを目標としている．Traceability Recovery は，多数の文書やソースコードが与えられた状態で，それらの 1 つを基点に，関連する他の文書やソースコードの集合を検索する技術となっている．

研究の実施における 1 つの課題が，手法の評価方法である．対応付け技術を検索問題として定式化すると，検索結果に含まれるべき「正解」を準備し，機能名などから検索した結果に正しい検索結果が含まれるかどうかを再現率と適合率により評価することができる．ただし，Traceability Recovery のように設計文書などを用いる技術の場合は，対外的に設計文書などが公開されていることは少ないため，実験結果を他の手法と比較することがむずかしいという制限もある．

他の評価指標としては，関数やメソッドを機能に関係していると思われる順に並べたとき，最上位の「正解」要素の順位を評価する Effectiveness Measure も提案されている[4)]．この指標は，モジュールの上位から順に結果を調査していく開発者の作業効率を表現しているが，開発者がメソッドの一覧を閲覧しても正解であるかどうかをただちに判定できるとは限らないという点で実際の効率と異なるとも指摘されている[3)]．

現状の評価方法では，いずれも検索の正確さを評価しているが，ソフトウェアごとにも結果は異なるため，実際のソフトウェア開発で利用可能であるかどうかという観点ではまだ不十分である．また，対象ソフトウェアの選定方法や正解データの作り方といった実験の妥当性，ツールの使いやすさ，検索対象である機能の指定しやすさや結果の閲覧しやすさなどの開発者の作業に即した評価の実施方法，実際の開発者への展開方法など，様々な課題が残されている．本セッションは，これらの様々な課題を整理し，参加者が相互に問題意識や技術的課題を理解する機会としたい．

参　考　文　献

1) Eaddy, M., Zimmermann, T., Sherwood, K. D., Garg, V., Murphy, G. C., Nagappan, N. and Aho, A.V.: Do Crosscutting Concerns Cause Defects?, *IEEE Transactions on Software Engineering*, Vol. 34, No. 4, pp. 497–515 (2008).
2) Gethers, M., Dit, B., Kagdi, H. and Poshyvanyk, D.: Integrated Impact Analysis for Managing Software Changes, *Proceedings of the 34th IEEE/ACM International Conference on Software Engineering*, pp.430–440 (2012).
3) Parnin, C. and Orso, A.: Are automated debugging techniques actually helping programmers?, *Proceedings of the 20th ACM International Symposium on Software Testing and Analysis*, pp.199–209 (2011).
4) Poshyvanyk, D., Guéhéneuc, Y.-G., Marcus, A., Antoniol, G. and Rajlich, V.: Feature Location Using Probabilistic Ranking of Methods Based on Execution Scenarios and Information Retrieval, *IEEE Transactions on Software Engineering*, Vol.33, No.6, pp.420–432 (2007).
5) Rajlich, V.: *Software Engineering: The Current Practices*, CRC Press (2012).
6) Rajlich, V. and Wilde, N.: The Role of Concepts in Program Comprehension, *Proceedings of the 10th International Workshop on Program Comprehension*, pp.271–278 (2002).