用途に応じたトレーサビリティ復元とその有効性の評価に向けて

城 間 祐 輝[†1] 風 戸 広 史[†1]

ソースコードと機能の対応関係を特定する技術の一つとしてトレーサビリティ復元がある．本稿では復元するトレーサビリティリンクの種類を概観し，用途に応じた復元すべきトレーサビリティリンクの種類や用途に対する有効性の評価方法について議論する．

Towards Recovering and Evaluating Effective Traceability Links Based on Their Usage

Yuki Shiroma[†1] and Hiroshi Kazato [†1]

Traceability recovery is one of the techniques that discover correspondence between program elements in source code and software features. In this position paper, we review the types and the usages of traceability links and discuss how we can apply and evaluate traceability recovery techniques according to the usages of the recovered traceability links.

1. はじめに

トレーサビリティリンク (Traceability Link; TL) とはソフトウェア開発における成果物間を関連付ける能力 (トレーサビリティ) を明示的に表したものである．成果物は要求や設計等の異なる抽象度，構造や振る舞いなどの異なる側面から記述されるため，多様な種類があり，それらの間にはさまざまな関連がある．Spanoudakis らは TL を次の 8 種類に分類した[1)]．

(1) *Dependency*: ある要素の変更がもう一方の要素に及ぶ
(2) *Generalisation/Refinement*: 複雑な要素を分解した，複数の要素を 1 つの要素にまとめた
(3) *Evolution*: 開発や保守の過程で，ある要素が別の要素に置き換わる
(4) *Satisfiability*: ある要素が別の要素を必要としている
(5) *Overlap*: 2 つの要素が同じシステムやドメインに関する機能を参照している
(6) *Conflict*: 2 つの要素が競合している
(7) *Rationalisation*: ある要素が，別の要素が生成された理論的根拠を表している
(8) *Contribution*: 要素の生成に貢献したステークホルダーとの関係

†1 NTT ソフトウェアイノベーションセンタ
NTT Software Innovation Center

トレーサビリティには様々な用途があり，用途に応じて利用する TL の種類が異なる．前述の Spanoudakis らは TL の用途を 1) 変更影響分析・変更管理，2) 標準準拠分析・システム検証，3) ソフトウェア成果物の理解，4) ソフトウェア成果物の再利用，の 4 種類に分類した．たとえば，変更影響分析では変更要求の対象機能が記述された開発文書からソースコードへの Overlap の TL を辿ることで変更すべきソースコード上の要素が特定できる．その後，特定したソースコード上の要素から Dependency の TL を辿ることで，変更の波及先である要素が特定できる．

このように，開発作業の支援のために TL を利用することは有益であるが，開発現場において成果物間のトレーサビリティが失われることがしばしばあるため，計算機を用いて TL の復元を支援する試みがある．多くの先行研究では情報検索 (Information Retrieval; IR) 技術を応用し，文書間に出現する単語の一致度を基に TL を復元する．この手法で復元した TL の種類は Overlap にあたり，異なる抽象度や側面にまたがる成果物において，同一の機能が出現する箇所を抽出することができる．トレーサビリティ復元手法の評価は，TL の正解集合を人手で作成し，それに対する再現率と適合率を用いることが一般的である．

しかし，先行研究では Overlap の TL を復元した後の用途を明示しておらず，TL の復元結果が開発作業の支援において有効であるかどうかを評価において考慮していない．トレーサビリティの用途によっては

表 1　用途に応じた TL の種類，復元方法，および評価方法の整理

用途	TL の種類	復元方法	TL の利用方法	評価方法
変更影響分析	Overlap	IR 技術	Overlap を用いて機能の出現箇所を発見し、Dependency を用いて変更の影響範囲を見積もる	修正箇所を正解集合としたときの再現率/適合率
	Dependency	静的/動的解析		
...	...	...	...	...

Overlap 以外の TL を復元して利用すべき場面もあると我々は予想している．

以上のことから，我々は用途に応じて復元する TL の種類や復元方法を決定し，用途に対する有効性を評価すべきであると考える．本稿ではワークショップにおける議論のため，それらを整理する方針を検討する．

2. 情報検索技術を応用したトレーサビリティ復元技術

Antoniol らは IR 技術を用いるトレーサビリティ復元手法を提案した[2]．Antoniol らの手法では 2 つの文書の記述に含まれる単語の一致の度合いを基に，文書間の類似度を算出し，類似度の高い文書を TL の候補として出力する．その後の多くのトレーサビリティ復元の研究においても同様の IR 技術が応用されている．

これらの先行研究の評価では TL の正解集合を手動で作成し，この正解集合に対してトレーサビリティ復元手法が出力した候補の再現率や適合率を求める．たとえば，ソースコードと開発文書間のトレーサビリティ復元手法を提案している Chen らの研究では，一定以上の開発経験のある複数人の分析者が次の 2 つの規則をもとに TL の正解集合を作成した[3]．

- 開発ドキュメント中の記述がクラス名やクラス中の識別子について直接言及している場合に TL があると判断する
- 開発ドキュメント中の記述において対象のクラスが責務を果たしていると考えられる場合に TL があると判断する

このような評価は，用途とは無関係に人手で正解集合を与え，それに対する TL 手法の出力の正しさを評価しており，TL の用途や目的が達成できるかという有効性は評価しない．このことから，我々は想定する用途が TL によってどの程度支援できるかを指標として手法の有効性を評価するべきだと考える．

3. トレーサビリティ復元技術の整理

用途ごとに利用する TL の種類，復元方法，評価方法について **表 1** の形で整理したい．たとえば，変更影響分析の用途では以下のように整理できる．

- **利用する TL の種類** 1 節で述べたように，Overlap と Dependency の TL を組み合わせて利用し，変更される要素とその影響が波及する要素の候補を出力する．
- **TL の復元方法** Overlap の TL は IR 技術の応用により復元できる．一方で，要素間の依存関係はソースコード中の単語よりもソースコードの構造や振る舞いに強く依存するため，Dependency の TL は IR 技術を応用するだけでは精度よく復元できない．ソースコードの静的解析や動的解析を用いて 2 つの要素が依存関係にあることを特定する必要があると考える．
- **用途に応じた TL の評価方法**変更影響分析の用途では，変更要求に対する実際の変更箇所が正解集合となる．この正解集合に対して，前述の方法で復元した TL の再現率と適合率を算出する．

4. おわりに

以上の考察をふまえて，ワークショップでは 1) TL の用途，2) TL の種類や用途に応じた復元方法や評価方法の 2 点について議論したい．

参考文献

1) Spanoudakis, G. and Zisman, A.: Software Traceability: A Roadmap, *Handbook of Software Engineering and Knowledge Engineering* (Chang, S.K., ed.), Vol.3, World Scientific Publishing Co., pp.970–983 (2005).
2) Antoniol, G., Canfora, G., Casazza, G., De Lucia, A. and Merlo, E.: Recovering traceability links between code and documentation, *IEEE Transactions on Software Engineering*, Vol.28, No.10, pp.970–983 (2002).
3) Chen, X. and Grundy, J.: Improving automated documentation to code traceability by combining retrieval techniques, *Proc. 26th IEEE/ACM International Conference On Automated Software Engineering (ASE'11)*, pp. 223–232 (2011).