# 修士学位論文

題目

# ソフトウェアトレーサビリティリンクの回復技術に関する系統的レビュー

指導教員

楠本真二

報告者

LYU WEIXUAN

令和 8 年 2 月 2 日

大阪大学 大学院情報科学研究科

コンピュータサイエンス専攻

令和 7 年度 修士学位論文

ソフトウェアトレーサビリティリンクの回復技術に関する系統的レビュー

LYU WEIXUAN

## 内容梗概

本研究は，ソフトウェアトレーサビリティリンク回復（Traceability Link Recovery: TLR）を対象に，2011 年から 2025 年に発表された関連研究に対して系統的文献レビュー（SLR）を実施し，「リンク種別・技術パラダイム・データセットの公開性」という三次元的視点から研究の全体像を整理・分析した．具体的には，体系的な検索とスクリーニングを経て 77 本の論文を選定し，以下の 4 つのリサーチクエスチョン（RQ）に取り組んだ．RQ1 ではリンク種別（Requirement–Code, Issue–Code 等）の分布，RQ2 では技術パラダイムの構成と時系列的進化（IR, ML, DL, LLM, その他），RQ3 では評価用データセットの公開性構造（公開のみ/非公開のみ/併用/自作），RQ4 では「リンク種別 × 技術パラダイム」「リンク種別 × データセット公開性」の組み合わせが示す構造的な選好と潜在的バイアスを明らかにした．

分析の結果，以下の知見が得られた．第一に，TLR 研究はリンク種別において顕著な集中を示し，Requirement–Code と Issue–Code が長期にわたり主要な比重を占めているが，両者の成長ペースや問題設定は異なる．第二に，技術パラダイムは時間軸上で単純な「新旧交代」ではなく，多パラダイムの並存を示している．IR は初期から優位性を持ち重要なベースラインとして機能し続ける一方，DL は 2017 年以降顕著に増加し，近年では事前学習モデルや LLM に関連する研究も出現している．第三に，データセットの公開性に関しては，全期間を通じて「公開データセットのみ利用」する研究が支配的であるが，リンク種別によって公開データへの依存度には差異が見られる．第四に，クロス分析により，主流リンク種別は特定の技術パラダイムおよび公開データと安定した組み合わせを形成しやすい一方，非主流リンク種別は非公開または自作データに依存する傾向が強く，その結果，比較可能性，および研究間の蓄積において構造的な制約に直面していることが明らかになった．

以上の知見に基づき，本研究は，研究の偏りを是正するための多様なリンク種別に向けた評価基盤の整備の必要性を指摘するとともに，TLR 研究の実務適用性を向上させるための今後の方向性について論じる．

### 主な用語

ソフトウェアトレーサビリティ, Traceability Link Recovery (TLR), 系統的文献レビュー (Systematic Literature Review)，情報検索 (Information Retrieval)，機械学習・深層学習 (Machine Learning / Deep Learning)，大規模言語モデル (LLM)

# 目次

# 図目次

## 表目次

## 1　はじめに

ソフトウェアトレーサビリティ（Software Traceability）は，高品質なソフトウェア工学を実現するための重要な能力として広く認識されている．要求，設計，コード，テスト，欠陥などの成果物間に利用可能な追跡関係が存在する場合，エンジニアリングチームは変更影響分析，コンプライアンス検証，カバレッジ分析などの主要な活動をより効果的に実施できる．特に航空や医療などのセーフティクリティカル分野において，トレーサビリティはしばしば認証や承認の要件と直接結びついている [2]．しかし，トレーサビリティは同時に，強く求められながらも真に定着させることが困難な工学的メカニズムであり，実践においては事後的な追跡や形式的な保守が頻繁に見られ，その価値を持続的に実現することが困難であるという現状がある [3, 4]．

上記のギャップが生じる核心的な原因の一つは，コストと複雑性にある．大規模プロジェクトにおける成果物の数量と可能なリンク数は急速に膨張し，手作業による追跡リンクの作成と保守にかかる工数はしばしば高すぎて現実的ではない．同時に，ソフトウェア開発の動的な進化は，リンクの保守を煩雑かつエラーが発生しやすいものにする．さらに，実際の開発環境において，成果物は異種ツールやシステム（異なるバージョン管理システム，要求/欠陥管理システム，ドキュメントリポジトリなど）に分散しており，本来的に互いに分断された状態にあるため，高品質なリンクを持続的に抽出し保守することは困難かつ高コストである [5]．

このような背景の下，研究界では多数の自動または半自動の追跡リンク確立技術が提案されてきた．その中で「トレーサビリティリンク回復（Traceability Link Recovery: TLR）」は長らく主流の用語として使用され，頻繁に情報検索（IR）問題としてモデル化されてきた．すなわち，成果物の自然言語コンテンツを用いて類似度を計算し，候補リンクを生成して関連順にランク付けし，分析者の確認に供する手法である [6]．この種の手法は，人手によるスクリーニングの負担を大幅に軽減できるが，精度不足の問題が繰り返し指摘されており，分析者は依然として結果の判定にかなりの時間を費やす必要がある [2]．さらに，成果物間の「意味論的ギャップ（Semantic Gap）や語彙の不一致」（同一概念が異なる成果物で異なる表現で記述されること）は，テキスト類似度に基づく手法の効果をさらに弱める要因となる [7]．

上述のボトルネックに対処するため，研究はより豊かな技術的アプローチを継続的に探索している．一方で，人間と計算機の協調によるフィードバックメカニズムを通じて IR のランキングを改善する試み（例えば，限られたユーザーフィードバックを伝播させ，未検証の候補リンクの順位向上に利用するなど）が行われている [8]．他方で，より意味表現能力の高い表現を学習し，語彙の不一致や意味論的ギャップを緩和するために，機械学習（ML）や深層学習（DL）が導入されている [9, 10]．近年の研究では，IR と事前学習済み言語モデルに基づく深層学習手法の体系的な比較も始まっており，特定の条

件下では深層学習アプローチが大規模プロジェクトにおいてより強力な性能ポテンシャルを持つことが報告されている [11].

同時に，既存の二次研究は以下の点を示唆している．TLR 研究は一定の規模に達しているものの，実験的評価は小規模なデータセットに依存していることが多く，産業現場での実証研究は依然として不足している．また，コンテキスト，ツールの詳細，用語の使用に関しても改善の余地があると報告されている [5].

以上の議論に基づき，本研究は「成果物・技術・データセット」の 3 軸を用いて，2011 年から 2025 年までの TLR 関連研究を体系的に整理することを目的とする．

具体的には，以下の 2 点を提供することを目指す．

1. **研究者への新たな研究の起点**: 既存研究の全体像を可視化し，研究者が断片的な試行錯誤を避け，より戦略的に次の研究課題を設定するための基盤を提供する．
2. **開発者への技術選択に関する俯瞰的な情報**:「どの成果物ペアに対して，どの技術が主流であり，どのようなデータセットで評価されているか」という現状と傾向を俯瞰的な情報として提供し，実務家がプロジェクトの特性に応じて適切な技術アプローチを選択するための判断材料を提供する．

本論文では，コミュニティで広く使用されている Traceability Link Recovery（TLR）の表記を踏襲し，研究対象を「既存の成果物間に，自動または半自動の方法で追跡リンクを確立（回復/生成）する技術およびその実証評価」に限定して分析を行う．

本論文の構成は以下の通りである．

- **第 2 章**: 関連用語と分類軸を整理し，研究範囲と境界を画定する．
- **第 3 章**: 系統的文献レビューの方法論，検索戦略，および選定プロセスについて説明する．
- **第 4 章**: レビュー結果を報告する．
- **第 5 章**: リサーチクエスチョンに基づいた詳細な議論を行い，研究への示唆を提示する．
- **第 6 章**: 全文を総括し，将来の研究方向を示す．

# 2　準備

## 2.1　ソフトウェアトレーサビリティの基礎概念

### 2.1.1　ソフトウェアトレーサビリティとは

ソフトウェアトレーサビリティは，ソフトウェア工学において重要でありながら実現が困難な能力として長く認識されてきた．特に安全重要システムにおいては，認証やコンプライアンス要件と結びつき，要求の実装証明，検証網羅性，変更影響の制御可能性などを実証するために期待されている．一方で，研究では，実践におけるトレーサビリティがコストや保守の負担により形骸化しがちであり，その「あるべき価値」を持続的に実現することが困難であることが指摘されている [12].

概念的なレベルにおいて，CoEST（Center of Excellence for Software and Systems Traceability）は，ソフトウェアトレーサビリティを以下のように定義している．

> 「ソフトウェア工学において，一意に識別可能な成果物を相互に関連付け，それらのリンクを経時的に維持し，リンクネットワークを利用して製品やプロセスに関する問いに答える能力」[2]

この定義は，トレーサビリティが単に「リンクの確立」にとどまらず，「継続的な保守」と「問題解決指向の利用」を含むものであることを明確にしている．

これに呼応するように，書籍『Software and Systems Traceability』は，より基礎的な視点から「トレース（trace)」をソース成果物，ターゲット成果物，および両者間のリンクからなる三項関係として記述し，トレーサビリティを「これらのトレースを確立し使用する能力または潜在性（Potentiality)」とみなしている．この表現は，トレーサビリティが静的なドキュメント産物ではなく，「関係ネットワーク」に向けた工学活動であることを強調している [2].

しかし，真の困難はそれを持続可能な工学的メカニズムとして定着させる点にある．Gotel らは，トレーサビリティが開発や保守を改善し得るとしても，自動化が不十分であれば，リンクの作成と進化的な保守に著しいコストがかかると指摘している．また，事後的なリンク回復と比較して，プロジェクトの知識が失われていない段階で早期に追跡関係を確立・固定化する方が経済的かつ高品質であることが多いが，現実にはチームメンバーの流動やシステムの進化が，保守を長期的な課題にしている [13].

したがって，本研究では TLR（Traceability Link Recovery）の議論に入る前に，本章において関連用語，範囲，および分類軸（成果物の種類，粒度，方向性など）を明確化する．その目的は，概念の混用による誤解を低減し，後続の結果提示と考察に対して一貫した解釈の枠組みを提供することにある．

具体的なイメージとして，ソフトウェア開発における各成果物の階層的な関係と，トレーサビリティリンクの方向性や次元の概念を図 1 に示す．本図は，抽象度の異なる成果物間（例：要求とコード）を

結ぶ垂直方向の追跡や，同レイヤー内での水平方向の追跡を視覚的に整理したものであり，本研究におけるリンク種別の議論の前提となる．

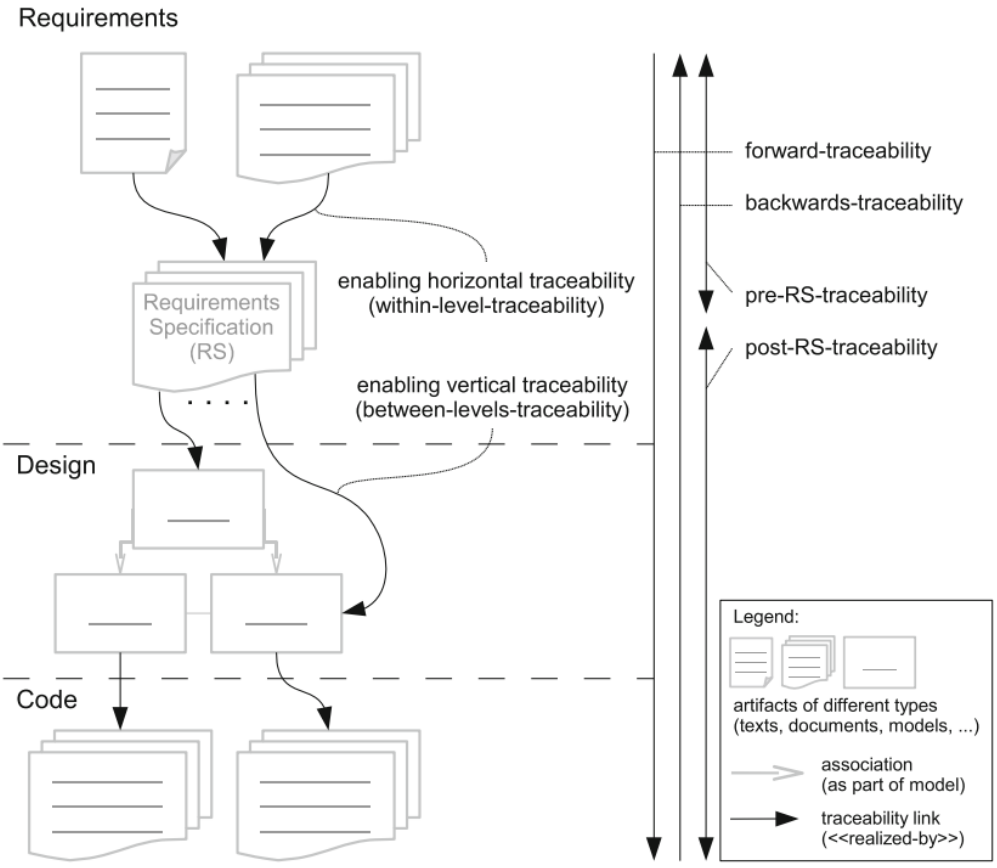


図 1　トレーサビリティリンクの次元と方向（文献 [1] より引用）

### 2.1.2　トレーサビリティリンクとは

「トレーサビリティリンク（Traceability Link）」はソフトウェアトレーサビリティの中核的な媒体であり，異なるソフトウェア成果物間に確立された，解釈および利用可能な関連関係を指す．要求工学やモデル駆動開発（MDE）に関するサーベイにおいて，追跡は通常，成果物間に解釈可能な関係を確立することによって実現される．これらの関係は文献上しばしば「トレーサビリティリンク」と呼ばれ，成果物間の依存関係，影響関係，因果関係などを記録するために用いられる [1]．

表現レベルにおいて，トレーサビリティリンクは方向性を持つ関係のエッジとして形式化できる．例えば，書籍『Software and Systems Traceability』に収録されている MDE 追跡情報モデルを例にとると，Traceability Link は「ソフトウェア成果物間の何らかの依存関係を具現化する」実体として定義さ

れ，その読み取り方向はソース（source）とターゲット（target）によってコード化される．また，リンクは「コンテキスト階層」（あるリンクが別のリンクの文脈内に存在するなど）を持つことも可能である [2]．これに対応して，Traceability LinkType は「タイプ層（type level）」において成果物タイプ間の依存意味論を規定し，ソースやターゲットなどの要素を通じて「どのようなタイプのオブジェクトがどのようなタイプのリンクによって接続され得るか」を記述する [2]．

### 2.1.3 TLR（Traceability Link Recovery）とは

本論文のタイトルは, 研究コミュニティで広く使用されている用語である Traceability Link Recovery（TLR）を踏襲している．既存研究において，TLR はしばしば「与えられた 2 組の既存ソフトウェア成果物に対し，それらの間のリンクを推論し返却する」タスクとして操作的にモデル化される．典型的な手法は，片側の成果物をクエリとみなし，情報検索手法を用いてもう一方の集合から候補を検索・ランク付けし，候補リストによって人間のリンク確認作業を支援するものである．また，より理想的な目標は，2 組の成果物間のすべての有効なリンクを直接返却することである [14]．

注意すべき点として，この「成果物間リンクを自動/半自動的に推論する」タスクに対し，文献上では多様な用語表記が存在する．例えば，ある研究では表記上 recovery や retrieval を用い，IR/ML 等の手法を通じて検証用の候補リンクリストを生成し，実践上のリンク確立プロセスを実現することを強調している [15]．その一方で，generate や creation などの表記を直接使用し，「生成された追跡リンク（generated trace links）」や「自動追跡リンク作成アルゴリズム（automated trace link creation algorithms）」について論じている研究も存在する [11]．

したがって，本論文のレビュー選定基準においては，「用語表記の一致」ではなく「タスクの本質的一致」という原則を採用する．すなわち，既存成果物を入力とし，成果物間の追跡リンクを自動または半自動的に生成/検索/回復/確立することを目的とするもの（例えば，候補リンクのランク付けリストの出力や，二値的なリンク判定など）は，すべて本研究でいう TLR の範疇とみなす．これにより，用語の違いのみに起因して関連研究を体系的に見落とすことを回避する．

## 2.2 成果物（Artifacts）とリンク種別

RQ1（既存の TLR 研究が主にどのような成果物間で追跡リンクを確立・回復しているか）に回答し，後続の RQ2（技術アプローチ）および RQ3（データセットの公開性）に対して一貫した統計基準を提供するため，本節ではまず概念レベルの分類枠組みを提示する．具体的には，本研究における「成果物（artifact）」の定義，採用する成果物タイプ，およびそれに基づく「リンク種別（link type）」の定義を明確にする．このアプローチは，一意に識別可能な成果物間にリンクを確立するというトレーサビリティ研究の基本的な設定と整合するものであり [2]，結果の集計に入る前に対象と関係の境界を統一す

ることで，後続のコーディングと計数の再現性を保証する．

本研究において，リンク種別は「成果物ペア（artifact pair）」を基本単位（例：Requirement–Code, Issue–Code など）とし，ある研究が「どの 2 種類の成果物間のトレーサビリティリンクを回復しようとしているか」を記述するために用いる．強調すべき点として，本節で提供するのは，第 4 章で提示するリンクマトリックスや論文分類を支えるための概念的な枠組みである．

### 2.2.1 リンク対象成果物の種類

**(1) 分類の原則と範囲** トレーサビリティリンクの前提は，リンクされる対象が工学活動において位置特定可能かつ区別可能（一意に識別可能）な成果物であることである [2]．したがって，成果物タイプの区分はすべての工学的形態を網羅することを目的とするのではなく，システマティックレビューのコーディング過程において，主流な研究対象をカバーし，かつ境界が明確で再現可能なカテゴリ体系を提供することを目的とする．これは，同一カテゴリの成果物が論文によって異なる名称や媒体で記述されることに起因する分類の不一致を低減するためである．

本研究では，TLR 研究における一般的な成果物を 6 大カテゴリ（Requirement，Design，Code，Test，Issue，Document）に抽象化する．この分類の合理性は以下の 3 点に基づく．

1. **中核的な工学チェーンへの対応**: Requirement–Design–Code–Test は，ソフトウェア開発ライフサイクル（特に V モデル）における中核的な成果物チェーンに対応しており，従来のトレーサビリティ議論および多くのリンク種別における最も直接的な帰着点であるため，分類の主幹として自然な妥当性を持つ [16].
2. **現代的実践との整合（Issue の台頭）**：アジャイルやツールチェーンの進化に伴い，Issue（Jira 等の ITS における issue/bug report/feature request など）は，実践において要求，欠陥，タスクなどの多重な意味を担っており，多くの追跡研究における「一級の成果物（First-class Artifact）」となっている．実際，Issue ベースのトレーサビリティに関する SLR においても，Issue を成果物の範疇に含めなければ，ITS を中心とする現代の開発実践と乖離が生じることが指摘されている [17].
3. **包括性と一貫性のバランス（Document の活用）**：「中核ではないが研究に確実に出現する」成果物を無理に上述の 5 カテゴリに含めることを避けるため，本研究では Document をカバーカテゴリとして設定した．これは，API ドキュメント，設計説明，ユーザーマニュアルなど，Requirement/Design/Test に安定して分類することが困難な文書型成果物を集約するためのものである．これにより，研究対象の多様性を反映しつつ，過度な細分化による一貫性の低下を防ぐことができる．

**(2) 6 種類の成果物の定義と例**　以下に各カテゴリの定義，具体例，および本研究における位置づけを示す．

1) **要求系成果物**（Requirement）

システムが備えるべき機能，制約，および品質要求を記述する成果物を指す．

- **例**: 要求仕様書（SRS），ユーザーストーリー，要求項目/文，ポリシー/規制条項など．
- **役割・範囲**:「システムが何をなすべきか」を規定する意味的なレイヤーとして位置づけられる．

2) **設計系成果物**（Design）

「要求をいかに実現するか」という構造や方針を記述する成果物を指す．

- **例**: アーキテクチャ設計書，モジュール仕様書，UML モデル（クラス図等），設計決定記録など．
- **役割・範囲**: 要求仕様と実装コードの間をつなぐ，媒介的な構造記述としての役割を担う．

3) **コード系成果物**（Code）

ソフトウェア実装の具体的なプログラム成果物を指す．

- **例**: ソースコードファイル，クラス，メソッド，関数など．
- **役割・範囲**: 実装層の証拠として扱われる．なお，バージョン管理上の「変更型成果物（コミットや Change Set）」も本カテゴリに含める．

4) **テスト系成果物**（Test）

実装が要求を満たしているかを検証し，品質を保証するための成果物を指す．

- **例**: テストケース，テストスクリプト，テスト手順書，テスト仕様書など．
- **役割・範囲**: 実装の正当性を確認する役割を持ち，検証カバレッジや影響分析の文脈で用いられる．

5) **イシュー系成果物**（Issue）

開発プロセスにおいて欠陥，改善，変更要求などを記録・管理するための成果物を指す．

- **例**: バグレポート
- **役割・範囲**: 保守フェーズにおける欠陥管理や，変更要求に伴う影響追跡の主要な対象となる．

6) **サポート文書類**（Document）

システムの理解，利用，または保守を支援するための補足的な文書を指す．

- **例**: ユーザーマニュアル，API ドキュメントなど．
- **役割・範囲**: 他の 5 カテゴリ（要求・設計・コード・テスト・イシュー）のいずれにも属さ

ない，支援的な説明資料として定義範囲を限定する．

### 2.2.2 リンク種別（artifact pair）の整理

本研究において,「リンク種別 (link type)」は，ある TLR 研究がどの成果物間にトレーサビリティリンクを回復しようとしているかを記述するために用いられる．トレーサビリティ研究における「一意に識別可能な成果物間にトレースリンクを確立する」という基本的な設定と整合させるため，本研究ではリンク種別を「成果物ペア（artifact pair)」として定義する．すなわち，2.2.1 項で定義した成果物タイプを 2 つ組み合わせることで得られるカテゴリレベルのリンク対象である（例：Requirement–Code, Issue–Code)．

**(1) 命名と表記：最小分析単位としてのカテゴリペア**　統一的なコーディングと統計を行うため，本研究では以下の表記規約を採用する．

- $A$–$B$:「成果物カテゴリ $A$ とカテゴリ $B$ の間のリンク種別」を表す．

本節（概念層）において，$A$–$B$ は無向のカテゴリの組み合わせとして扱う．リンクの方向性（例えば $A \to B$ や $B \to A$）はリンクの属性に属し，後続の小節で個別に定義・コーディングする（2.2.4 項参照)．

なお，1 本の論文が複数のリンク種別を同時に研究している場合は,「出現すればカウントする」という原則に基づき，複数のリンク種別として記録する．この「まずカテゴリペアでリンク種別を定義し，方向性を独立した属性として扱う」という処理方法は，トレースリンクが「主要な追跡方向（primary direction)」を持ちうるという記述と整合する一方で,「どのようなカテゴリの組み合わせが存在するか（RQ1)」と「それらの組み合わせが通常どのような方向でタスク設定されているか（方向性分析)」を区別して回答することを容易にする．

**(2) リンク種別の意味的解釈**　以下の 15 種類のリンク種別は，本研究の分類枠組みにおける「概念的に可能な集合」である．これらの組み合わせの中には，既存研究で頻繁に見られるもの（例：Requirement–Code，Issue–Code）もあれば，比較的稀少または研究が分散しているものもある．本章では意味的な解釈のみを示し，統計的結論については第 4 章で述べる．

- **Requirement–Code**: 要求とソースコード間のリンク回復．実装のカバレッジ検証や影響分析などに用いられる．
- **Issue–Code**: Issue（欠陥/変更要求等）とコード間のリンク回復．実装修正箇所の特定や保守支援などに用いられる．

- **Requirement–Design**: 要求と設計間のリンク回復．設計の要求カバレッジ確認や，要求変更に伴う設計箇所の特定などに用いられる．
- **Requirement–Requirement**: 要求間のリンク回復．主に上位要求（High-level）と下位要求（Low-level）間の階層的な追跡に用いられる．
- **Design–Code**: 設計とコード間のリンク回復．実装が設計構造や意図に適合しているかの検査に用いられる．
- **Code–Test**: コードとテスト間のリンク回復（どのモジュール/関数がテストされているかなど）．回帰テストの選択や故障箇所特定によく用いられる．
- **Requirement–Test**: 要求とテスト間のリンク回復．要求のテストカバレッジ検証や，要求駆動のテスト管理などに用いられる．
- **Design–Test**: 設計とテスト間のリンク回復．主要な設計要素/シナリオがテストされていることを保証するために用いられる．
- **Code–Document**: コードと文書間のリンク回復（API ドキュメントとコード要素の対応，説明文書と実装の対応など）．
- **Design–Document**: 設計と文書間のリンク回復（アーキテクチャ/設計説明と関連支援文書の対応など）．
- **Issue–Issue**: Issue 間のリンク回復（バグレポート間の相互関連など）．
- **Test–Document**: テストと文書間のリンク回復（テスト仕様/説明と文書項目の対応など）．
- **Requirement–Document**: 要求と文書間のリンク回復（要求とユーザーマニュアル/運用説明/ガイドライン項目の対応など）．
- **Issue–Design**: Issue と設計間のリンク回復．欠陥/変更を設計レベルの根拠や影響範囲にマッピングするために用いられる．
- **Issue–Test**: Issue とテスト間のリンク回復．

#### 2.2.3 粒度の整理

本研究の目的は，TLR 研究における「リンク対象の組み合わせ」,「技術アプローチ」,「データセットの公開性・入手源」などの巨視的な特徴を体系的に整理し，リサーチクエスチョンに回答するとともに再現可能なマッピング結果を構築することにある．

リンクの粒度に関して，既存研究では粒度の選択がコスト対効果のトレードオフと強く関連していることが指摘されている．例えば，要求をメソッドレベルまで追跡することは，クラスレベルへの追跡に比べて著しく高い投入コストを要する一方で，そのリンク品質が必ずしも優れているとは限らない [2]．さらに，現行の TLR 手法の中には，「内部的には微細粒度の類似度を用いつつ，出力としては粗粒度の

リンクを返す」という戦略をとるもの [7] や，多層的な粒度を併用する情報統合メカニズムを持つもの [18] が散見される．このような状況下において，システマティックマッピングの中で「粒度」を統一的にコーディングすることは，基準の一貫性と研究間の比較可能性を保証する上で困難である．以上の理由により，本研究では粒度を主要な統計次元ではなく，議論のための背景要因として位置づける．

以下に，各成果物タイプにおいて既存研究で一般的に見られる粒度レベルを示す（あくまで概念整理であり，統計的な結論を含むものではない）．

- **Requirement**: 要求ドキュメントレベル／章節レベル／単一要求（要求文，要求項目）レベル
- **Design**: 設計ドキュメントレベル／モジュールまたはコンポーネントレベル
- **Code**: ファイルレベル／クラスレベル／メソッド（関数）レベル
- **Test**: テストスイートレベル／テストケースレベル
- **Issue**: イシュー項目レベル
- **Document**: ドキュメントレベル／段落レベル

#### 2.2.4 境界事例

**(1) 複数成果物/マルチホップ（multi-hop）シナリオの扱い** 一部の研究では，2 種類の成果物間で直接リンクを回復するのではなく，中間成果物を介してマルチホップパス（例：Requirement → Design → Code）を構築することで，中間成果物を利用して用語のギャップを低減したり検索品質を向上させたりしている [19]．このような研究において，各ホップ（hop）は確立・検証・利用が可能な独立した追跡関係に対応しており，工学的実践においても独立したトレーサビリティリンクとして保守・消費されることが一般的である．

したがって，本研究でリンク種別をコーディングする際は，「明示的に出現するリンクごとに個別にカウントする」というルールを採用する．すなわち，ある研究が明示的に $R$–$D$ と $D$–$C$ という 2 区間の関係を構築している場合，Requirement–Design と Design–Code という 2 種類のリンク種別としてそれぞれ記録する（中間リンクも統計に含める）．この処理の目的は，研究が実際に回復したリンク対象の範囲を忠実に反映すること，およびマルチホップ手法を単一のエンドツーエンドのリンク種別として「畳み込む」ことによって，その方法論的特徴（中間成果物の関与自体が重要な設計選択であること）が失われるのを避けることにある．

**(2) 用語の不一致（recovery / retrieval / generation / establishment 等）の包含ルール** トレーサビリティ分野において，論文によっては「追跡リンクの確立または補完」という自動化プロセスを記述するために，recovery，retrieval，generation，establishment，maintenance などの異なる用語が使用されており，表記の揺れが存在する．

本研究の TLR に対する作業定義（2.1.3 項参照）に基づき，本研究では用語自体を包含・除外の基準とはせず,「タスクの実質」によってレビュー範囲に含まれるか否かを判定する．すなわち，論文の目標が自動または半自動的な手法を通じて，既存の成果物から「確認または利用可能なトレーサビリティリンク（通常は候補リンクリストやマトリックス形式で出力される)」を生成・回復・確立することにある限り，当該論文は本レビューにおける TLR 研究の範疇にあるとみなし，分析対象に含める．これにより，用語の違いのみに起因して関連研究を体系的に見落とすことを回避する．

## 2.3 TLR 技術の代表的アプローチ

既存の TLR 研究に対して一貫したコーディングを行うため, 本研究では「技術アプローチ（approach)」を，その問題モデリングの手法と主要な計算メカニズムに基づいていくつかの代表的なパラダイムに分類する．なお，本節では後続の RQ2 の統計を支えるための概念定義と判定ルールのみを提示し，具体的な手法の詳細や論文レベルの例示については第 4 章 4.3.3 項で述べる．

### 2.3.1 IR ベース（IR-based）

IR ベースの手法は，通常 TLR を検索ランキングタスクとしてモデル化する．すなわち，クエリ成果物（query artifact）が与えられたとき，候補集合との関連性を計算し，ランク付けされたリストまたは候補行列を出力する．本研究では,「類似度・関連性を核とし，ランキングリストを直接出力すること」を主要な特徴とする研究を IR ベースに分類する．

### 2.3.2 古典的機械学習（Classical ML）

古典的機械学習（Classical ML）のパラダイムは, 通常，リンク回復をリンク存在判定（link/non-link）または学習ランキング（Learning to Rank）問題として定式化する．これらは，ラベル付きリンクペアに基づいて分類器やランキングモデルを学習し，手動または自動構築された特徴量（テキスト，構造，メタデータなど）を利用して予測を行う．本研究では,「教師あり学習を核とし，明確な学習フェーズとラベルデータへの依存が存在する」研究を Classical ML に分類する．ただし，モデル主体がニューラルネットワークである場合は 2.3.3 項に分類する．

### 2.3.3 深層学習（DL）

深層学習（DL）のパラダイムは，ニューラルネットワークを用いて成果物間のベクトル表現（埋め込み表現：embedding）を学習し，マッチング・ランキング・判別を行う．通常，従来の特徴量エンジニアリングへの依存を弱め，エンドツーエンド学習や事前学習表現の転移を強調する．

本研究では,「深層ニューラルネットワークを主要なマッチャーまたは判別器として用いる」研究を

DL に分類する．

- **判定基準**: CNN/RNN/Transformer 等のアーキテクチャを用い，ラベル付きデータによる教師あり学習やファインチューニング（Fine-tuning）を経て分類モデルやランキングモデルを構築する手法を対象とする．
- **注記**: BERT や RoBERTa などの事前学習済み言語モデル（PLM）を使用する場合でも，それらをエンコーダとして利用し，下流タスクのデータセットでファインチューニングを行うアプローチは，本カテゴリ（DL）に分類する．

#### 2.3.4 大規模言語モデル（LLM-based）

LLM ベースの手法は，大規模言語モデル（LLM）を核とし，プロンプト（prompt）やコンテキスト構築を通じて候補リンクの判定，説明，または候補関係の生成を行う．

**分類に関する注記**: 技術分類の階層構造上，LLM は広義の深層学習（DL）のサブセットに位置づけられる．しかし，本レビューでは以下の理由により，LLM ベースの手法を独立したカテゴリとして扱う．

1. **アプローチの質的相違**: 従来の DL（2.3.3 項）が主に「学習データによるモデルパラメータの更新（Fine-tuning）」を前提とするのに対し，近年の LLM 手法は「プロンプトエンジニアリング」や「文脈内学習（In-Context Learning: Zero/Few-shot）」を主たる推論メカニズムとしており，その運用形態が根本的に異なるため．
2. **分析上の有用性**: この新しいパラダイム（生成 AI ベース）が TLR 分野に与える影響，特有の課題（ハルシネーション等），およびデータセット依存度の変化を個別に分析するため．

したがって，本研究では「LLM が最終的なリンク判定・生成に関与し，プロンプトや文脈内学習を中核的な推論モジュールとしている」研究を LLM ベースに分類する．

#### 2.3.5 その他（Other）

TLR の研究では，IR・（深層）学習といった主流以外にも，多様な問題設定に応じた非典型的アプローチが提案されている．本研究では，こうした多様な手法群を便宜的に “Other” として整理する．代表例として以下が挙げられる．

1. ルール／特徴条件に基づくリンク同定
2. 多層（カスケード）型の検出戦略
3. 進化計算による探索
4. 論理推論エンジンを用いた推論型アプローチ

### 2.4　系統的レビューの概要

本節では, 本研究が採用する方法論的基盤である系統的文献レビュー（Systematic Literature Review: SLR）の定義および一般的な実施プロセスについて概説する.

#### 2.4.1　系統的レビューとは

系統的文献レビュー（SLR）は，明確かつ再現可能な方法を用いて，ある研究課題に関連する既存の研究証拠を特定，評価，および統合する二次研究の一形態として一般的に定義される．その核心的な目的は，関連するすべての研究を可能な限り網羅した上で，監査可能で厳格，かつバイアスを最小限に抑えたプロセスを通じて，研究テーマに対する公平な評価を提供することにある [20].

ソフトウェア工学の文脈において，Kitchenham らは，SLR は単なる「手当たり次第の文献レビュー」ではなく, 特定の研究課題（RQ）に基づき,「入手可能なすべての関連研究」を体系的に特定, 評価, および解釈するものであると強調している [20]．彼らは SLR を二次研究として分類し，一次研究（primary study：実証研究）と対比させるとともに,「範囲の棚卸し」に重点を置くシステマティックマッピングとも区別している [20].

情報システム分野のガイドラインにおいて，Okoli & Schabram はさらに，厳密な系統的レビューは単なる「要約」にとどまらず，理論と証拠の統合および学術的な批判を含むべきであると強調している．また,「入力—処理—出力」のプロセスを十分に透明化し，他者が同一のプロトコル下で再現できるようにする必要があると述べている [21].

同時に,「レビューの書き方」という観点から，Webster & Watson は，質の高いレビューは著者中心（author-centric）ではなく概念中心（concept-centric）であるべきだと論じている．すなわち，レビューの組織構造は概念やテーマによって駆動されるべきであり，著者の要約を順に羅列するだけでは真の統合を実現することは困難であるとしている [22].

#### 2.4.2　一般的な実施プロセス

系統的レビューは通常,「計画—実行—報告」の 3 つの大段階を含む．Kitchenham らが提案したソフトウェア工学向け SLR ガイドラインでは，プロセスを計画（planning），実施（conducting），報告（reporting）に明確に区分し，プロセスの信頼性を保証するために監査可能な方法を用いることを強調している [20].

より詳細なレベルでは，Okoli & Schabram が「実行可能な段階的プロセス」を提示しており，その中には再現性を保証するために特に重要な以下の工程が含まれる [21].

1. **レビュー目的/研究課題の明確化**: まず，レビューが何を回答すべきか，および期待される出力

（証拠基盤の確立や総合的評価など）を明確に定義する．

2. **プロトコルの策定と一貫性の保証**: 文書化された実行可能なプロトコルを作成する．複数の研究者が関与する場合は，スクリーニングやコーディング基準の安定性を確保するためのトレーニングと一貫性チェックが強調される．
3. **検索と文献の特定**: 検索の体系性と網羅性を重視し，「馴染みのあるジャーナルや一部のソースのみを検索する」といったバイアスを最小限に抑える．
4. **実践的スクリーニング（包含・除外の判定）**: 検索された文献に対し，タイトルやアブストラクト等に基づく包含・除外基準を適用し，詳細な検討に進む文献を選別する．
5. **品質評価とスクリーニング**: 低品質な研究を除外する基準を明確にし，採用された研究の品質をスコアリングまたは記録することで，後続の統合プロセスの入力品質を保証する．
6. **データ抽出**: 事前に定義されたフィールドに従って体系的に情報を抽出し，一時的かつ恣意的な項目の増減によるバイアスを回避する．
7. **統合と分析**: 抽出された事実を適切な方法（定量的，定性的，またはその組み合わせ）で統合する．核心は「証拠の集合」から解釈可能な結論構造を形成することにある．
8. **執筆と再現可能な報告**: 読者が「入力—処理—出力」の流れを理解し，原則としてレビュー結果を再現できるように，プロセスを十分に詳細に記述する．

さらに，Webster & Watson は「検索の閉ループ」に直接関連する操作上の提案を行っている．彼らは，主要な論文を初期特定した後，参考文献を遡る後方追跡と被引用文献を追う前方追跡を行い，文献集合を拡張して網羅性を高めることを推奨している．また，読解から統合への移行を支援するために「概念マトリックス」の使用を提案している [22].

### 2.4.3 関連分野の具体例

本分野において「系統的レビュー」が具体的にどのように実施されているかをより明確にするため，本節ではソフトウェアトレーサビリティおよびそのサブトピックに関連深く，かつレビュー構造が完全な代表的先行研究をいくつか対照例として挙げる．これらの例は，その結論をすべて再述するためのものではなく，リサーチクエスチョンの設定方法と範囲の画定方法を説明するために用いる．

**例 1：「トレーサビリティ研究の全景」を指向したレビュー** Winkler と von Pilgrim のレビューは，「要求工学およびモデル駆動開発（MDD）におけるトレーサビリティの研究と実践」を対象としている．まず基本用語と定義（例えば IEEE 用語集によるトレーサビリティの定義や，トレース，成果物，トレーサビリティリンクなどの核心概念の議論）から入り，続いて関連研究をテーマごとに体系的に分類し，さらに実践上の制約と将来の課題について議論している [1].

**例 2：「特定技術（NLP/LLM）によるトレーサビリティ支援」を指向したレビュー**　Guo らの章は「要求トレーサビリティにおける NLP」を主線としている．まず要求トレーサビリティの定義と工学活動の分解（計画/作成/保守/利用）を示し，その上で研究対象を最も頻繁に研究されるタスクの一つである Trace Link Recovery (TLR) に絞り込んでいる．彼らは TLR を「ソース集合とターゲット集合が与えられたとき，追跡関係を満たす二項関係集合を回復する」という形式的なタスクとして表現し，TLR が実践において重要である理由の一つとして「リンクが体系的に作成・保守されていないか，経時的に失われるため，事後的な回復が必要である」点を指摘している．手法に関しては，IR,（浅層）機械学習，深層学習，および近年の LLM へと至る系譜的な紹介を行っている [23].

**例 3：「新興成果物形態（Issue)」を指向した体系的文献レビュー**　Lyu らの SLR は「イシューレポートを核心成果物とする要求トレーサビリティ」に特化している．彼らは,「従来の要求トレーサビリティのレビューはしばしばイシューを一級の成果物として扱っておらず，現代の ITS（Jira など）主導の開発実践と乖離がある」と明確に指摘している．そのため，明確な検索と選定プロセスを用いて大量の候補文献から関連研究を抽出し,「問題，成果物ペア，技術，評価対象」などの次元に沿って RQ に回答している [17].

### 2.5　小括

本章では，系統的レビューの実施前提として，TLR 関連概念と研究対象に関する予備的な整理を行った.

第一に，ソフトウェアトレーサビリティ，トレーサビリティリンク，および TLR の基本的な定義と研究範囲を明確にし，後続の研究課題（RQ）に対する統一的な用語基盤を提供した．第二に，本研究のコーディングに用いる成果物タイプ，リンク種別，および方向性などの重要な次元を提示し，マルチホップ追跡や用語表記の不一致といった境界事例の処理ルールについて説明した．最後に，系統的レビューの一般的な定義と実施プロセスを概説し，関連分野の代表的なレビュー事例と照らし合わせることで，本研究の方法選択の妥当性と位置づけを説明した.

これらの準備作業は，第 3 章のレビュー実行プロセスおよび第 4 章の結果提示に向けた，再現可能な分類枠組みと方法論的基礎を確立するものである.

# 3　実施した系統的レビュー

本章では，本研究で実施した体系的文献レビューのプロセスと詳細について説明する．特に「どのように検索し，どのように選定し，どのようにコーディングと集計を行ったか」という全過程を明示することで，研究の透明性と再現性を高めることを目的とする．なお，コーディングとは，各論文から抽出した情報を，事前に定義した分析用カテゴリに体系的に割り当て，後続の統計的集計や分析に利用可能な形式へ変換する作業を意味する．後続の章との役割分担は以下の通りである．第 4 章では主に統計とクロス分析の結果を提示し，第 5 章ではそれに基づいた解釈的な議論を展開する．一方，本章は方法論の章として，これらの結果を支えるプロセスの根拠と判断ルールを提供し，結論が単なる経験的な集約に留まらないようにする．

本研究は，ソフトウェア工学分野で一般的に用いられる体系的レビューの実践（例えば，Kitchenham らが提案した計画・実行・報告の基本指針など）を参照し，レビュープロセスを以下の 5 つの段階に構成した．

1. リサーチクエスチョン（Research Questions）と範囲の定義
2. 検索戦略の設計と文献収集
3. 包含・除外基準の設定と段階的選定
4. 情報抽出と分類コーディング
5. 統計的統合と結果提示の準備

これらの段階は相互に独立しているわけではない．例えば，試行検索や一次選定の過程で明らかになったノイズや曖昧な項目は，検索式や除外基準の修正を促す．また，コーディング過程で現れる境界事例（同一研究が複数の成果物や追跡関係を含む場合など）は，統一された判定ルールによって標準化する必要がある．本章では，後続の分析が一貫した基準に基づいていることを保証するため，これらの重要なプロセスについて明確に記述する．

具体的には，3.1 節でまず本研究のリサーチクエスチョン（RQ）とレビューの境界を定義する．3.2 節では，検索期間，データソース，検索式の構築原則を含む検索戦略について述べる．3.3 節では，選定フローと包含/除外基準を示し，重複排除と段階的選定の手順を説明する．3.4 節では，情報抽出フォームの項目設計とコーディングスキームについて説明する．最後に 3.5 節で，統計と統合の方法を概説し，それらが第 4 章の結果提示にどのように対応するかを述べる．この構成により，本章は追跡可能かつ再確認可能な研究プロセスの記述を提供し，読者が論文集合から統計結果に至る生成経路を理解し再現できるようにする．

### 3.1 リサーチクエスチョン（Research Questions）の設定

体系的文献レビューの核心は，明確なリサーチクエスチョン（RQ）を導き手としてレビュー対象と情報抽出の範囲を限定し，既存研究の単なる羅列に陥ることを避ける点にある．本研究は，トレーサビリティリンク回復研究を対象とし、「リンクの種類、技術的アプローチ、データソースの公開性およびそれらの相互関係」を中心的な分析軸として設定する。その上で、既存の近年における TLR 研究の全体像を多角的に把握し、後続の総合的な考察の基盤を提供することを目的として、四つの研究課題（RQ1〜RQ4）を設定する。

**RQ1：既存の TLR 研究はどの種類の成果物間のリンク確立を目的としているか？** RQ1 は，近年の TLR 研究が扱う「追跡対象は何か」，すなわちどのソフトウェア成果物間にリンクを確立または回復しようとしているかに焦点を当てる．これに答えるため，本研究ではまず成果物を統一的に類型化（Requirement，Design，Code，Test，Issue，Document など）し，その上で論文が関与するリンクタイプ（Requirement–Code，Issue–Code など）を特定する．RQ1 の結論は，「研究の網羅性」と「研究の集中度」の 2 つの側面，すなわちどのリンクタイプが頻繁に研究され，どれが比較的稀少であるかという問いに答えるために用いられる．

**RQ2：TLR を実現するために，どのような技術的アプローチが用いられ，それらは時間と共にどう進化してきたか？** RQ2 は，近年の TLR 研究が「どのような技術を用いて実現しているか」，およびそれらの技術的アプローチに経時的な進化の傾向が存在するかどうかに焦点を当てる．これに答えるため，対象論文の中核的な手法を分類（情報検索 IR，機械学習 ML，深層学習 DL，大規模言語モデル LLM に基づく手法，およびその他の戦略など）し，出版年と組み合わせて統計を行うことで，TLR 分野における各技術アプローチの相対的な割合と段階的な変化を明らかにする．

**RQ3：TLR 研究において，使用されるデータセットの公開性や入手源にはどのような傾向があるか？** RQ3 は，近年の TLR 研究が「どのようなデータに依存しているか」，特にデータセットの公開性と入手源が研究の再現性と一般化可能性に与える影響に焦点を当てる．これに答えるため，論文で使用されたデータセットに着目し，その公開レベルを分類（公開データセットのみ利用，非公開/商用データセットのみ利用，公開・非公開の併用，自作データセットなど）する．これにより，後続の分析において，異なるリンクタイプや技術アプローチがデータソースへどのように依存しているかを分析することが可能となる．

**RQ4：TLR 研究の成果物，技術，データセットの間にはどのような関係性や課題が存在し，将来の方向性は何か？** RQ4 は，次元を横断した総合的な問いに焦点を当てる．すなわち，異なる成果物リンク

タイプ（RQ1），技術アプローチ（RQ2），データセットの入手源（RQ3）の間に体系的な結合関係が存在するか，そしてその結合関係がどのような研究上の制約，バイアス，あるいは将来の機会をもたらすかという点である．これに答えるため，本研究ではさらにクロス集計（RQ1 × RQ2，RQ1 × RQ3 など）を行い，「リンクタイプ—技術アプローチ—データセット公開性」の複合的な視点から研究構造を特定し，現在の TLR 研究における主要な課題と潜在的な研究方向性を帰納的に要約する．

本研究の全体的な構成と各リサーチクエスチョン（RQ）の有機的な関係性を図 2 に示す．本研究では，この枠組みに従い，第 4 章において RQ1 から RQ3（成果物・技術・データセットの各次元の独立分布）および RQ4（次元間の相互関係と構造的偏り）の統計結果を提示する．続く第 5 章では，これらの量的結果に対して解釈的な議論を行い，その形成要因，研究の空白，および将来の発展方向についてさらに分析を行う．

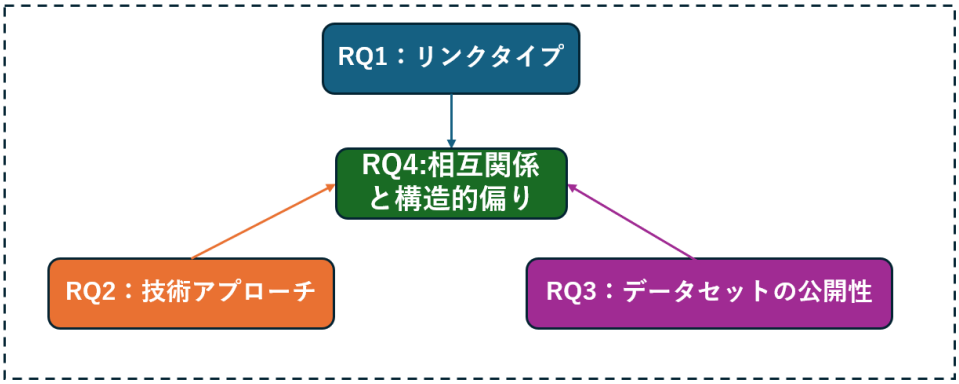


図 2　本研究のリサーチクエスチョンと分析の枠組み

### 3.2　検索戦略

本節では，文献収集の具体的な戦略と実行設定について述べる．これには検索期間，データソース，検索式の構築原則，および補完的検索（スノーボール/手動検索）の方法が含まれる．プロセスの再現性を保証するため，本研究では各データソースの検索フィールド，選定条件，実行日を可能な限り明確に記録し，検索結果を後続の選定プロセス（3.3 節）の入力とする．

#### 3.2.1　検索期間

本研究では，検索対象期間を 2011 年から 2025 年に設定した．この期間設定の主たる理由は，TLR や情報検索の思想的起源を遡ることではなく，現在の研究と実践に対して最も説明力を持つ技術的進化の段階に分析の焦点を合わせることにある．

第一に，TLR 技術は過去 10 数年の間に，従来の IR ベースの手法から，機械学習，深層学習，そし

て近年の LLM へと，研究パラダイムが再構築される過程を経てきた．もし開始点をさらに過去へ遡らせた場合，古典的な IR を中心とする研究（タスク設定や評価方法が現在と大きく異なるもの）が数量的に支配的となり，統計結果に構造的なバイアスが生じる恐れがある．これは結果として「過去の主流技術」を過剰に反映させ，現代に向けた技術進化の軌跡を正確に描写することを妨げる．したがって，2011 年を起点とすることで，IR から（機械/深層）学習，そして LLM への段階的な進化と手法の多様化傾向をより鮮明に提示し，後続の章における技術的転換点と研究トレンドの変化に関する比較分析を有効に機能させる．

第二に，本研究は現在の研究活動に対して直接的な示唆を提供することを目的としている．あまりに古い時期の研究は，現代のツールチェーン，データ規模，開発プロセスとは明らかに異なる前提に基づいていることが多く，その結論の転用可能性や現実的な適用性は限定的である．時間枠を直近の約 15 年間に限定することは，分野の進化を反映するのに十分な範囲を確保しつつ，対象研究間のデータ・工学環境・評価パラダイムにおける比較可能性を高め，レビューの結論が「現代的な文脈」においてより高い説明力と参考価値を持つようにするためである．

### 3.2.2 データソース

TLR 研究が主にソフトウェア工学および情報検索に関連するジャーナルや会議で発表されることを考慮し，本研究では以下のデータソースを選択して体系的な検索を行った．

- IEEE Xplore
- ACM Digital Library
- SpringerLink
- ScienceDirect

### 3.2.3 検索語と設計方針

本研究の主検索語は，TLR 分野で広く用いられている代表的表現 “traceability link recovery” とした．これは，トレーサビリティ研究において同一概念に対する表現揺れが大きく（例：recovery / generation / establishment / identification 等），同義語を OR で網羅的に展開すると検索結果が過度に膨張し，関連性の低い文献（ノイズ）が急増するためである．実際，予備検索の段階で同義語を広く含む検索式を試行したところ，特定のデジタルライブラリでは検索精度が著しく低下し，会議・論文種別でフィルタリングを施しても実務的にスクリーニング可能な規模を超える候補集合が生成されることを確認した．

一方で，短いフレーズのみに依存する場合，表現差による漏検のリスクが残る．そこで本研究では，

主検索フレーズによる取得結果を起点とし，引用追跡（前向き／後向きスノーボール）を併用して漏れを補完する戦略を採用した．具体的には，主検索により得られた候補集合を段階的にスクリーニングし，全文確認により確定した論文群をシード（seed）として，以下の 2 つの追跡を実施した．

1. シード論文の参考文献リストに基づく後向き追跡
2. シード論文の被引用文献に基づく前向き追跡

これにより，主検索フレーズとは異なる語彙で記述されている関連研究も回収可能である．

さらに，漏検リスクを評価するため，必要に応じて “traceability link generation” や “Traceability Link Creation” 等の近接表現を用いた限定的な確認検索を実施し，主要な関連研究が引用追跡により回収されていることを確認した．

### 3.3　研究論文の選別

#### 3.3.1　選定プロセスの全体像

本研究では，データベース検索で得られた候補文献に対し，形式基準および内容基準による段階的スクリーニングを実施し，さらに参照・被引用の追跡（前向き／後向きスノーボール法）によって漏れを補完した．選定プロセスの全体像と各段階の件数推移を図 3 に示す．

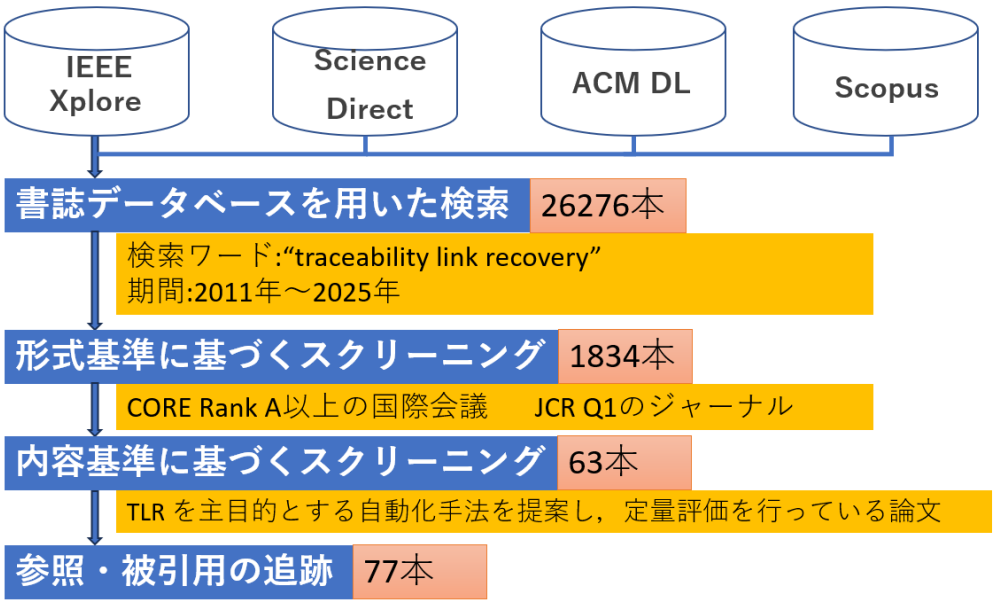


図 3　選定プロセスの全体像

### 3.3.2　形式基準に基づくスクリーニング

データベース検索（期間：2011–2025，検索語：“traceability link recovery”）により得られた候補集合（図 3 参照）に対し，まず出版媒体の品質を担保する目的で形式基準による絞り込みを行った．具体的には，国際会議は CORE RankingsPortal においてランクが A 以上，ジャーナルは JCR Q1 を満たすものに限定し，会議・ジャーナル以外の出版形態を除外した．

この形式基準により，初期候補 26,276 件から 1,834 件へと候補集合を縮減した．データベース別の内訳を表 1 に示す．

表 1　形式基準によるスクリーニング後のデータベース別内訳

| データベース | 件数 |
|---|---|
| IEEE Xplore | 192 |
| ACM Digital Library | 1,383 |
| SpringerLink | 54 |
| ScienceDirect | 205 |
| **合計** | **1,834** |

### 3.3.3　内容基準に基づくスクリーニング

次に，形式基準を満たした文献（1,834 件）に対し，研究目的・貢献が本研究の対象に合致するかを確認するため，内容基準に基づくスクリーニングを行った．内容基準は以下の観点から設定した．

1. 研究の主目的がトレースリンクの自動生成・復元・更新にあること．
2. 再現可能な自動化手法（IR/ML 等）を提案していること．
3. Precision 等の指標による定量的検証を実施していること．
4. 対象成果物が具体的であり，実質的なソフトウェア成果物間リンクを扱うこと．
5. 論文の中心的貢献が TLR 手法の提案・評価に置かれていること．

スクリーニングは二段階で実施した．まずタイトル・要旨に基づき対象適合性を判定し，次に本文確認により「手法提案と定量評価が中心貢献であるか」「リンク対象が具体的成果物であるか」を最終確認した．加えて，複数データベースから重複取得された文献は DOI・タイトル等に基づき統合し，重複を除外した．

その結果，最終的に 63 件を一次選定文献として確定した（図 3 参照）．

### 3.3.4　参照・被引用の追跡による補完

主検索フレーズに基づくデータベース検索は高精度である一方，分野固有の表現揺れにより漏れが生じ得る．そこで本研究では，一次選定文献（63 件）をシード（seed）とし，後向き追跡（参考文献）および前向き追跡（被引用）を併用して追加文献を探索した．追跡で得られた候補についても，3.3.3 項と同一の内容基準で採否判定を行った．

その結果，引用追跡により 14 件を追加し，最終的な対象文献数は 77 件となった（図 3 参照）．

## 3.4　データ抽出とコーディング

本研究の RQ1 から RQ4 に回答するため，最終的に選定された 77 本の文献を精読し，統一された抽出フォームに基づいて各文献の重要な情報を記録した．その後，抽出結果を統計可能なカテゴリ変数にマッピングし，第 4 章の分布統計および第 4.5 節のクロス分析（RQ1 × RQ2，RQ1 × RQ3）に使用した．

本研究の抽出項目は「成果物—技術—データセット」の 3 次元構造を中心に設定されており，主に以下の項目を含む．

- **基本情報**: 論文タイトル，出版年，発表チャネル（会議/ジャーナル）．
- **リンクタイプ（Link Types, RQ1）**：論文が対象とする成果物タイプおよびそのリンクタスク．論文間の一貫性を保証するため，追跡対象となる成果物を 6 つのカテゴリ（Requirement / Design / Code / Test / Issue / Document）に統一して統合し，それに基づいて論文がカバーする具体的なリンクタイプ（Requirement–Code，Issue–Code など）を特定した．
- **方向性（Direction-Intent, RQ1 方向拡張）**：各リンクタイプに対し，著者がタスクレベルで設定した「Source → Target」の方向を記録し，$A \to B$，$B \to A$，無向/双方向の 3 種類に統一してコーディングした．
- **技術アプローチ（Approaches, RQ2）**：論文が TLR を実現するために採用した主要な技術カテゴリを記録し，年次比較やリンクタイプ間の比較を支援するために，IR / ML / DL / LLM / Other の 5 種類に統合した．
- **データセットの公開性（Dataset Openness, RQ3）**：データソースと入手可能性を記録し，「公開データセット / 非公開（産業または商用）/ 公開・非公開の併用 / 自作データセット」等のカテゴリに従ってコーディングした．

上記の基準に基づいて，実際に選定された論文に対してコーディングを行った具体例を図 4 に示す．

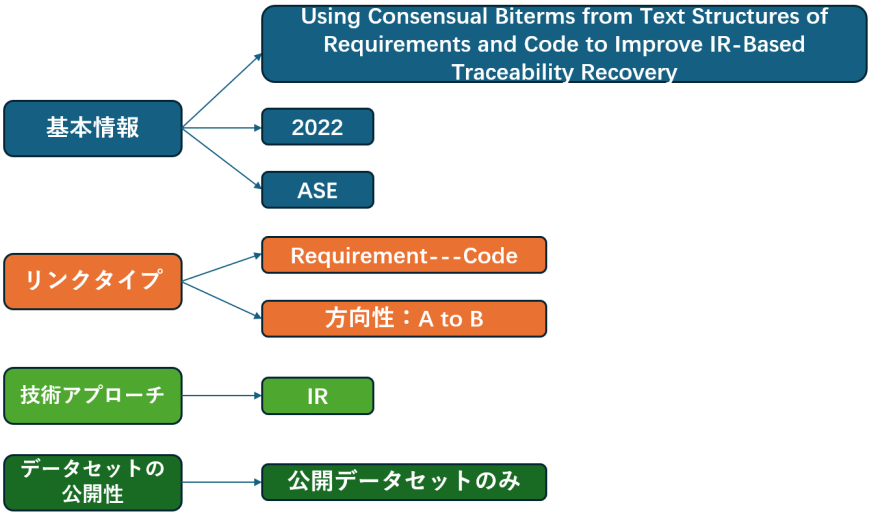


図 4　1 つの論文に対するコーディングの適用例

### 3.5　データ合成と分析方法

文献の抽出とコーディングの完了後，RQ1 から RQ4 への体系的な回答を支援するため，コーディングデータの合成を行った．本節では，第 4 章「レビュー結果」における図表と結論の厳密性を確保するために，統計対象，指標設計，および可視化・提示形式について説明する．

#### 3.5.1　RQ に対応する統計対象と指標設計

用語やタスク表現の論文ごとの差異による比較の歪みを避けるため，本研究では分析段階において「コーディング後のカテゴリ変数」を統一的な統計対象とし，それらを各リサーチクエスチョンにマッピングした．

- **RQ1（リンクタイプ）**：6 種類の成果物（Requirement / Design / Code / Test / Issue / Document）を基礎とし，以下の項目を統計する．
  1. 各成果物ペア間のリンクタイプ分布（マトリックス形式での提示）．
  2. 1 本の論文がカバーするリンクタイプ数の分布（研究対象の複雑さ/範囲を描写するため）．
  3. 各リンクタイプの方向性分布（ $A \to B$ / $B \to A$ / 無向または双方向）．
  4. 代表的なリンクタイプの時間的分布（リンクタイプごとの年次分布）．
- **RQ2（技術アプローチとその進化）**：各論文の主要手法を IR / ML / DL / LLM / Other の 5 つに統合し，各リンクタイプにおける技術カテゴリの出現状況を統計するとともに，出版年情報と組み合わせて技術路線の進化傾向を記述する．

- **RQ3（データセットの公開性）**：データセットの入手源と可用性に基づき「公開のみ / 非公開のみ（産業または商用）/ 併用 / 自作」にコーディングし，異なるリンクタイプにおける分布を統計する．これは再現性と研究参入障壁の問題を議論するために用いる．
- **RQ4（次元間の関係と制約）**：RQ1 から RQ3 の結果に基づきクロス合成を行い，特に RQ1 × RQ2 と RQ1 × RQ3 の 2 つのクロス統計表を構築し，「リンクタイプ—技術アプローチ—データの可用性」の間に一定の関係が存在するかどうかを明らかにする．

### 3.5.2　クロス分析（RQ4）の構築方法

RQ4 に回答するため，本研究ではリンクタイプを列次元とし，以下の 2 種類のクロス表を構築した．

- **RQ1 × RQ2（リンクタイプ × 技術アプローチ）**：各リンクタイプに対し，IR / ML / DL / LLM / Other の各技術カテゴリでの出現回数を集計する．これは，異なるリンクタスクが安定した手法の選好を示しているかを観察するために用いる．
- **RQ1 × RQ3（リンクタイプ × データセット公開性）**：各リンクタイプに対し，「公開のみ / 非公開のみ / 併用 / 自作」などのデータソースカテゴリでの出現回数を集計する．これは，異なるリンクタスク間での再現性，データ取得の障壁，産業データへの依存度の差異を議論するために用いる．

なお，クロス表の結果は因果関係の説明と直接等価ではないことを強調しておく．したがって，第 5 章の議論パートにおいて，クロス集計結果に対するさらなる解釈と限界分析を行う．

# 4　レビュー結果

## 4.1　調査対象論文の外観

第 3 章で述べた一連の選定プロセスを経た結果，最終的に 77 本の文献を分析対象として確定した．本章の以下の節では，この文献セットに基づき，各 RQ に対する分析結果を順次報告する．

## 4.2　RQ1 への回答

選定された 77 本の論文を分析し，各論文が回復または追跡対象としているすべてのリンクタイプを抽出した．集計に先立ち，本研究では文献ごとに異なる用語を統一的に扱うため，対象となるソフトウェア成果物を以下の 6 つのカテゴリに分類・定義した．

- **要求系成果物 (Requirement)**: ソフトウェアに求められる機能や振る舞いを定義する成果物
- **設計系成果物 (Design)**: 要求をどのように実現するかの構造や方針を示す成果物
- **コード系成果物 (Code)**: ソフトウェアを構成する具体的なプログラムコード
- **テスト系成果物 (Test)**: 実装された機能の品質を検証・保証するための成果物
- **イシュー系成果物 (Issue)**: 開発中に発生したバグや変更要求などを追跡・管理する成果物
- **サポート文書 (Document)**: 他の成果物の理解や利用を助けるための補足的な文書

上記の 6 種類の成果物定義に基づき，抽出されたリンク関係を分類した結果，合計 15 種類の具体的な追跡リンクタイプが特定された．特定された 15 種類のリンクタイプと，それらを扱っている具体的な論文およびその本数を表 2 に示す．

### 4.2.1　研究対象リンク種別の全体像

表 3 は，特定されたリンクタイプの詳細な分布状況を示している．特に留意すべき点として，統計手法は論文単位ではなく，リンクインスタンス（Link Instance）単位のカウントを採用している．すなわち，1 本の論文が複数のリンクタイプ（例えば，Requirement-Code と Requirement-Design の両方）を研究している場合，それぞれのカテゴリに個別にカウントしている．そのため，表中のカウント総和は論文総数（77 本）を超過している．

表 3 から，研究対象となるリンクタイプの分布には明らかな集中傾向が見られる．出現頻度が最も高いのは Requirement–Code（30 件）と Issue–Code（20 件）であり，次いで Requirement–Design（15 件），Design–Code（11 件），Test–Code（10 件），Requirement–Test（9 件）となっている．

リンクタイプの数が多岐にわたるため，以下では出現頻度が最も高い 3 つのリンクタイプに焦点を当て，その研究動機と適用シナリオについて簡潔に説明する．

表 2　リンクタイプごとの関連論文と本数

| リンクタイプ | 含まれる論文 | 本数 |
|---|---|---|
| Requirement – Code | [15, 24, 6, 8, 25, 14, 3, 26, 27, 28, 29, 30, 19, 31, 32, 33, 7, 34, 35, 36, 37, 38, 39, 4, 40, 41, 42, 43, 44, 45] | 30 |
| Issue – Code | [40, 46, 11, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63] | 20 |
| Requirement – Design | [64, 25, 14, 10, 65, 29, 19, 31, 32, 33, 35, 66, 67, 9, 68] | 15 |
| Requirement – Requirement | [69, 14, 70, 29, 31, 32, 33, 71, 9, 72, 73, 74] | 12 |
| Design – Code | [14, 29, 75, 19, 31, 32, 33, 35, 4, 76, 77] | 11 |
| Code – Test | [18, 14, 78, 29, 19, 31, 32, 33, 79, 80] | 10 |
| Requirement – Test | [25, 14, 27, 29, 19, 31, 32, 33, 73] | 9 |
| Design – Test | [14, 19, 31, 32, 37, 81] | 6 |
| Code – Document | [82, 83, 30, 35] | 4 |
| Design – Document | [30, 35] | 2 |
| Issue – Issue | [84, 85] | 2 |
| Test – Document | [86] | 1 |
| Requirement – Document | [35] | 1 |
| Issue – Design | [87] | 1 |
| Issue – Test | [88] | 1 |

表 3　リンクタイプの分布マトリックス

| | **Requirement** | **Design** | **Code** | **Test** | **Issue** | **Document** |
|---|---|---|---|---|---|---|
| **Requirement** | 12 | 15 | 30 | 9 | 0 | 1 |
| **Design** | 15 | 0 | 11 | 6 | 1 | 2 |
| **Code** | 30 | 11 | 0 | 10 | 20 | 4 |
| **Test** | 9 | 6 | 10 | 0 | 1 | 1 |
| **Issue** | 0 | 1 | 20 | 1 | 2 | 0 |
| **Document** | 1 | 2 | 4 | 1 | 0 | 0 |

**(1) Requirement – Code**　本カテゴリは 30 本の論文を含み，TLR 分野で最も集中的に研究されている領域である．主な背景として，自然言語で記述された要求とプログラミング言語で記述されたコードの間には顕著な「セマンティックギャップ（Semantic Gap)」が存在することが挙げられる [7]．同一の概念を記述していても，両成果物はしばしば異なる用語（Vocabulary Mismatch）を使用するため，純粋な IR 類似度ランキングの有効性が著しく損なわれ [44]，リンク回復が極めて困難になる．以下に代表的な 2 つの研究を紹介する．

- **Ali らによる研究（セマンティックギャップと用語不一致の解決）[45]**: 要求文書とソースコード間の用語不一致（Vocabulary Mismatch）の問題に対し，この研究は，静的なテキストマッチングだけでは深い意味的関連を捉えることが困難であると指摘した．そこで研究者らは「Trustrace」と呼ばれる手法を提案し，ソフトウェアリポジトリ（Software Repositories）内の過去の変更データを「中間の架け橋」として利用した．履歴コミット記録に含まれる関連情報をマイニングすることで，Requirement-Code 間の独特な意味的断絶を効果的に埋め，リンク回復の精度を大幅に向上できることを示した．
- **Lucia らによる研究（ソースコードのテキスト疎性の改善）[36]**: Requirement-Code リンクにおけるソースコード側のテキスト情報不足（Data Sparsity）の問題に対し，コード内の識別子は通常短く分散しており，詳細な要求記述と直接マッチングさせることが難しいと指摘した．これに対し，コード特有の構造関係（制御フローやデータフローなど）を利用し，スムージングフィルタ（Smoothing Filter）を通じて関連するコード要素のテキスト情報を伝播・融合させる手法を提案した．この手法は，コードの構造的特徴を用いてテキスト記述を強化することで，Requirement-Code タイプの追跡精度向上に非常に有効であることが示された．

**(2) Issue – Code**　本カテゴリは 20 本の論文を含み，Requirement–Code に次ぐ第 2 の研究方向である．Issue（欠陥/要求項目）と Commit（コード変更）間の追跡リンクは，プロジェクトの進化履歴を記録し，プログラム理解，品質評価，欠陥予測などの保守活動を直接支援する．しかし実際のプロジェクトでは，この種のリンクは頻繁に欠落または未記録であり，保守の難易度を高め，プロセスの追跡可能性を損なっている [58]．欠落の原因は「リンク作成が強制ではない」ことや，開発の作業負荷や人為的ミスに関連している [59]．さらに，コミットメッセージに基づいて構築された「ゴールドスタンダード」自体も不完全である可能性があり（例えば，かなりの割合のコミットがいかなる Issue ID にも関連付けられていない)，自動回復の必要性がさらに強調されている [11]．

- **Bai らによる研究（ソーシャルプログラミング環境下の複雑な関連マイニング）[63]**: GitHub 環境における Issue と Pull Request (PR) のリンク予測問題に対し，従来のテキスト類似度に

基づく手法は，オープンソースコミュニティにおける豊富な「コンテキスト構造情報」を見落としていると指摘した．Issue と PR は孤立して存在しているのではなく，開発者，リポジトリ，タグなどのノードを通じて複雑な異種ネットワークを形成している．著者は異種グラフ学習（Heterogeneous Graph Learning）に基づく手法を提案し，これらの潜在的な構造化知識を捉えることを目指した．現代のオープンソース開発モデルにおいて，プロジェクト周辺のメタデータ（Metadata）を活用することで，テキスト記述のみに頼るリンク復元の限界を効果的に補完できることを証明した．

- **Sun らによる研究（セキュリティ脆弱性レポートの精密な特定）[55]**: ソフトウェアサプライチェーンセキュリティにおける課題，すなわち公開された脆弱性レポート（NVD/CVE 記述など）から具体的な「影響を受けるファイル（Vulnerability-Relevant Files）」をいかに迅速に特定するかという問題に対し，この特殊な Issue-Code タスクの難点を検討した．脆弱性の記述は通常抽象的であり，プロジェクト内部の具体的なコンテキストを欠いているため，直接的な特定は極めて困難である．著者は LLM と検索エンジンを組み合わせたフレームワークを提案し，専門家の検索推論プロセスを模倣することでこの難題を解決しようとした．この研究は，セキュリティという高リスクな垂直領域において，外部知識ベースと高度な自然言語理解技術を組み合わせることが，「脆弱性-コード」追跡問題の解決に不可欠であることを示した．

**(3) Requirement – Design**　本カテゴリは，要求と設計間のリンク確立に焦点を当てている．中心的な動機は，モデル駆動開発（MDD）において，自然言語要求をより洗練された構造・振る舞い表現へ「翻訳」する際，その根拠を追跡することにある．このプロセスは時間がかかりミスが発生しやすく，特に問題記述が長い場合，モデラーが「なぜそのようにモデル化したか」の根拠を遡ることは困難である．そのため，モデリング決定を双方向に追跡することは，モデルの完全性（Completeness）と簡潔性（Conciseness）のリスク管理に有利であるとされる [68]．また，安全クリティカルシステムにおいては，安全要求が設計モデルに反映されていることを保証し，認証機関の要求を満たすために不可欠である．設計レビューにおいて，検査者が特定の安全要求に関連する重要な設計フラグメントに集中できるよう支援する役割も果たす [65]．さらに，より広義のソフトウェア追跡の観点から見ると，要求-設計間等の成果物間リンクは，高信頼性システムにおいて認証機関によって構築と維持を要求されることが多いが，人手によるリンク作成は時間がかかりミスが発生しやすく，現実にはリンクが不完全または不正確である可能性もある．この現状もまた，自動化・半自動化支援への継続的な研究を後押ししている [10]．

- **Saini らによる研究（ドメインモデリングの決定）[68]**: 自然言語の問題記述からドメインモデル（クラス，属性，関係など）への追跡に注目した．要求記述に指示の曖昧さが存在する場合や，モデルに要求には明示されていない「暗黙の概念」が導入された場合，追跡リンクを通じてこれ

らのモデリング決定を明示化する必要性を強調した．これにより，モデルが完全かつ簡潔であるかを検証し，モデルから元の記述への根拠の逆引きを支援する．

- **Alenazi らによる研究（安全要求 ↔ 状態ベース設計モデル）[65]**: 安全クリティカルシステムを対象とし，安全要求を SysML ステートマシンなどの状態型設計モデルに追跡する手法を扱った．安全要求の追跡は，検査・保証・認証などの重要プロセスを支え，レビュー担当者が特定の安全要求に関連する状態や遷移などの設計要素に注意を集中させるのに役立つことを指摘した．

### 4.2.2　1 本の論文が扱うリンク種別数の分布

図 5 は，選定された 77 本の論文において，各論文が対象とした追跡リンクの種類数（リンクタイプ数）の分布を示している．

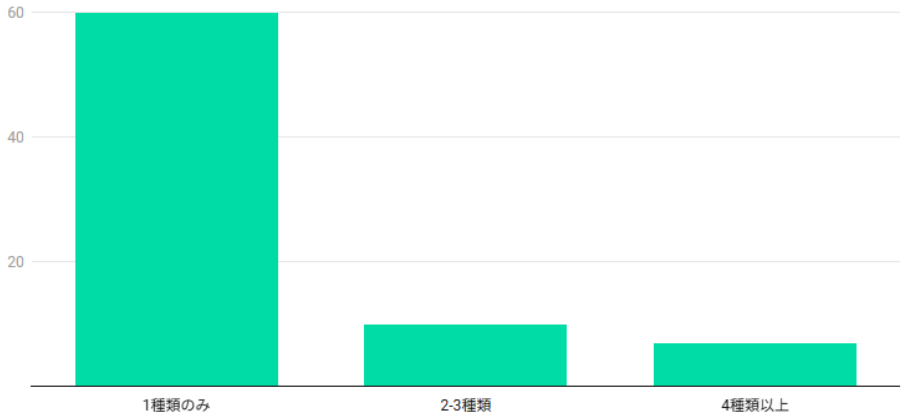


図 5　各論文が扱うリンク種別数の分布

統計結果によると，1 種類のリンクタイプのみを研究対象とした論文が最も多く，合計 60 本（全体の約 77.9%）を占めている．対照的に，2–3 種類のリンクタイプを扱った論文は 10 本（約 13.0%），4 種類以上のリンクタイプを同時にカバーした研究は 7 本（約 9.1%）であった．

全体として，既存の研究は単一のリンクタイプに焦点を当ててタスクを設定する傾向にあり，多種多様なリンクタイプを網羅する研究の割合は低いと言える．

この傾向の背景には，リンク種別ごとに成果物の特性（抽象度や記述言語）が大きく異なるため，汎用的な手法を構築するよりも，特定のリンクペアが抱える固有の課題（例：要求とコード間の語彙の不一致）の解決に特化する方が，研究としての貢献を明確にしやすいという事情があると考えられる．

#### 4.2.3 方向性の集計

**判定ルール** 本研究では，方向性を「リンク回復・生成タスクにおいて研究者が想定するデフォルトのクエリフローの向き」と定義する．すなわち，「ある成果物を入力（query/source）として与え，それに関連する別の成果物を出力（target）として検索または予測すること」を指す．この定義は，論文が仮定する典型的なワークフローを記述することを目的としており，利用段階でリンクが双方向に追跡可能かどうかを問うものではない．

コーディングの一貫性を保証するため，方向性の判定は以下の優先順位ルールに従って行った．

1. **目標記述の優先**: 論文が目標，問題定義，またはタスク記述において「given $A$, find $B$ / link $A$ to $B$ / retrieve $B$ for $A$」といった表現を明確に採用している場合，方向性を $A \to B$ と判定する．ここで，$A$ はクエリ端/起点成果物，$B$ は被検索端/目標成果物とする．
2. **タスク設定と評価プロセスからの推測**: 論文が目標記述で方向性を明言していない場合でも，手法のフロー，実験設定，または評価プロセスにおいて，ある成果物を入力とし，別の成果物を出力としていることが明確な場合（例：要求をクエリとし，コードファイルをランク付けしてTop-k 評価を行うなど），その実際のタスク設定に基づき方向性を推測し，入力端 $\to$ 出力端と判定する．
3. **「手法の対称性」は「双方向の意図」と等価ではない**: 一部の論文は，議論において「手法は両方向に適用可能である」と述べているが，実験や評価は単一方向のみをカバーしている場合がある．このような場合，本研究では論文の主要なタスク設定/評価方向を「Direction-Intent」として採用し（すなわちルール 1 または 2 に従って単方向と判定する），「理論的に交換可能であること」を「研究意図が双方向であること」として誤ってカウントし，統計結果を希釈することを避ける．
4. **双方向/無向（Bidirectional/Undirected）の判定条件**: 以下のいずれかの条件を満たす場合にのみ，方向性を双方向または無向と判定する．
   (a) 論文が目標または問題定義レベルで双方向タスクを明確に提起している場合（例：両端のユーザーを同時に対象とする，または $A \to B$ と $B \to A$ の両タスクを定義している）．
   (b) 論文が実験/評価において両方向の設定または報告を行っている場合．
   (c) 論文が方向性のない「依存/関連リンク集合」を回復するものであり，タスク設定から固定されたクエリ起点を特定できない場合．

上記のルールを通じて，方向性の統計結果は，研究コミュニティが異なるリンクタイプにおいて実際に注目しているデフォルトのワークフローの方向をより安定して反映でき，方向性の偏りとその要因に関する後続の議論の基礎を提供する．

**各リンク種別の方向性分布**　表 4 は，15 種類のリンクタイプにおける方向性の分布を集計したものである．統一的な形式で提示するため，各リンクタイプに対して列名「$A \leftrightarrow B$」を定義し，記号を以下のように定める．

- $A \to B$: $A$ 類の成果物をクエリ端/起点とし，それに関連する $B$ 類の成果物を出力端として検索または予測することを示す．
- $B \to A$: その逆方向を示す．
- **無向または双方向**: 論文が無方向の依存関係を回復しているか，あるいは目標記述/実験設定において両方向をカバーしているため，単一方向に分類できない場合を示す．

ここで強調すべき点は，$A$ と $B$ の具体的な意味はリンクタイプによって変化するということである．したがって，表 4 の目的は，リンクタイプごとに方向性の分布を観察することにあり，$A \to B$ や $B \to A$ を異なるリンクタイプ間で直接比較するための統一的な方向ラベルとして用いるものではない．

表 4　各リンクタイプにおける方向性の分布

| **リンクタイプ** ($A \leftrightarrow B$) | $A \to B$ | $B \to A$ | **無向 or 双方向** |
|---|---|---|---|
| Requirement – Code | 24 | 0 | 6 |
| Issue – Code | 13 | 2 | 5 |
| Requirement – Requirement | 8 | 0 | 4 |
| Requirement – Design | 9 | 0 | 6 |
| Design – Code | 3 | 2 | 6 |
| Code – Test | 0 | 6 | 4 |
| Requirement – Test | 5 | 0 | 4 |
| Design – Test | 2 | 0 | 4 |
| Code – Document | 1 | 2 | 1 |
| Design – Document | 0 | 1 | 1 |
| Test – Document | 0 | 0 | 2 |
| Issue – Issue | 2 | 0 | 0 |
| Requirement – Document | 0 | 0 | 1 |
| Issue – Design | 0 | 1 | 0 |
| Issue – Test | 1 | 0 | 0 |

統計データから，以下の顕著な傾向が観測できる．

1. **サンプル数が多いコード関連リンクタイプでは，「非コード側から実装側成果物を特定する」タスク設定が主流である．**
   - **Requirement ↔ Code**: Requirement → Code をタスク設定とする研究は 24 本であり，Code → Requirement を主とする研究は観測されなかった．
   - **Issue ↔ Code**: Issue → Code をタスク設定とする研究は 13 本であり，Code → Issue（2 本）や無向/双方向（5 本）も少数存在する．
   - **Code ↔ Test**: このタイプでは Test → Code を設定とする研究が 6 本（表中 $B \to A$）であるのに対し，Code → Test は 0 本であった．これは，テストとコード間のリンク回復において，テスト側から出発して対応する/カバーするコード実体を特定するタスクフローが一般的であることを示している．

   総合すると，これらの代表的なコード関連リンクにおいて，研究タスクは要求/Issue/テストをクエリ起点とし，コード（またはコード変更）を検索対象とすることが多く，結果として「実装側の特定（Implementation Localization）」という研究の偏りを示している．
2. **コードを含まない上流成果物間では，リンク回復の方向は主にソフトウェア開発のウォーターフォールモデルのトップダウンな流れに従う．**
   - **Requirement ↔ Design**: Requirement → Design を設定とする研究は 9 本であり，Design → Requirement は観測されなかった．
   - **Requirement ↔ Test**: Requirement → Test を設定とする研究は 5 本であり，Test → Requirement は観測されなかった．

#### 4.2.4 年次分布

**集計基準** 本節では，論文の出版年を時間軸とし，各リンクタイプ（link type）の年次分布を集計する．1 本の論文が複数のリンクタイプを同時に扱う可能性があるため，本節では「リンクタイプごとのカウント」を採用する．すなわち，1 本の論文が複数のリンクをカバーしている場合，対応する各リンクタイプにおいてそれぞれ 1 回ずつカウントする．注意点として，異なるリンクタイプのカウントを単純に合算して「年間の論文数」と解釈することはできない．本節の目的は，研究の関心点の時間的変化を描写することにある．

**主要リンクタイプの年次分布** 本節では，サンプル数が最も多い 2 つのリンクタイプである Requirement–Code と Issue–Code を取り上げ，その年次分布を示す（図 6，図 7）．

1. **Requirement–Code の年次分布**

   図 6 に示すように，Requirement–Code 関連の研究は時間軸上で広く分布しており，2011 年

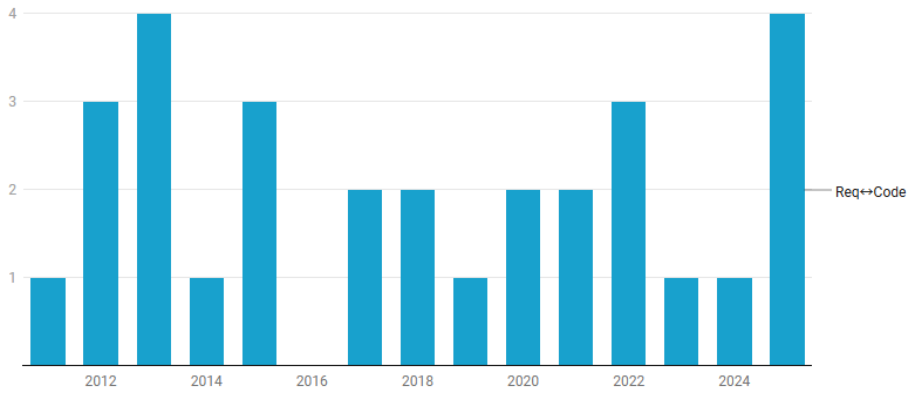


図 6　Requirement–Code リンクタイプに関する研究の年次推移

から 2025 年までの長期間をカバーし，強い継続性を示している．具体的には，該研究は初期（2011–2015 年）に一定の規模を形成している：2011 年（1 本），2012 年（3 本），2013 年（4 本），2014 年（1 本），2015 年（3 本）（計 12 本）．2016 年に一時的な空白が見られた後，2017 年から再び連続して出現している：2017–2018 年（各 2 本），2019 年（1 本），2020–2021 年（各 2 本），2022 年（3 本），2023–2024 年（各 1 本），そして 2025 年には 4 本に達している．全体として，Requirement–Code は初期の集中期と，その後の長期的かつ継続的な研究の蓄積の両方を含んでいる．

このような長期的かつ活発な研究の背景には，Requirement-Code 間のリンクがソフトウェア開発の実務において最も基礎的かつ不可欠な検証基盤であるという事情がある．具体的には，要求が正しく実装されているかを確認する受入テストや，機能変更時の影響範囲分析，さらには安全重要システムにおける認証取得において，このリンクの確立は必須となる．そのため，現場からの自動化需要が常に高く，研究対象として継続的に選択され続けていると推察される．

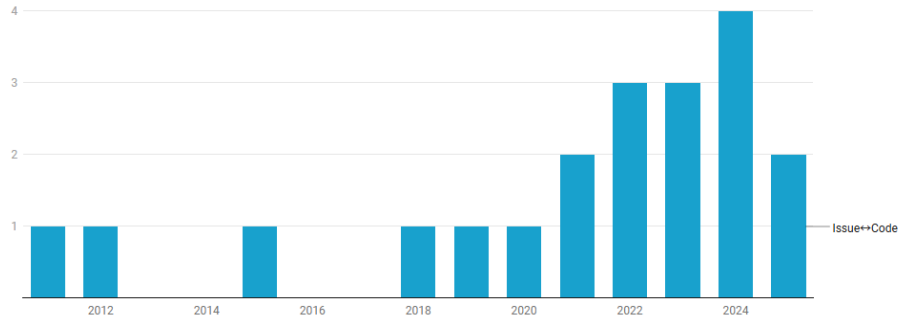


図 7　Issue–Code リンクタイプに関する研究の年次推移

2. **Issue–Code の年次分布**

図 7 によると，Issue–Code の研究は出現時期が比較的遅く，近年においてより顕著な集中傾向を示している．初期段階では散発的な出現に留まっている：2011 年（1 本），2012 年（1 本），2015 年（1 本）．その後，2018–2020 年の間は低頻度で推移した（各年 1 本）．これと比較して，2021 年以降の増加は顕著である：2021 年（2 本），2022 年（3 本），2023 年（3 本），2024 年（4 本），2025 年（2 本）．言い換えれば，Issue–Code の研究は主に直近 5 年間（2021–2025 年）のタイムウィンドウに集中しており（計 14 本），明らかな近年の集中という特徴を示している．

この近年の急増の背景には，深層学習（DL）技術の発展が大きく寄与している．Issue-Code 間のリンク回復は，自然言語による記述とソースコードという「異質なデータ間の意味的結合」を必要とするタスクであり，文脈理解や表現学習に長けた DL 技術がこの課題に対して極めて高い親和性を持つためである．なお，なぜ Issue-Code タスクにおいて特に DL が技術的に適合するのか，その構造的な理由の詳細については第 5 章にて議論する．

**小括**　主要な 2 つのリンクタイプを比較すると，以下のことが分かる．Requirement–Code は時間的次元においてより長い期間をカバーし，分布が連続的である．一方，Issue–Code は近年において強い集中と増加傾向を示している．

### 4.3　RQ2 への回答

#### 4.3.1　概要と技術分類

RQ2（技術手法の変遷）に回答するため，本研究では各論文で提案または採用された中核的なリンク回復技術（Core Recovery Technique）を調査した．77 本の論文で使用されている技術は多岐にわたるが，そのアルゴリズムの特性や歴史的背景に基づき，本研究ではそれらを以下の 4 つの主要なパラダイムおよび「その他」の計 5 つのカテゴリに分類した．各カテゴリの定義と含まれる代表的な手法は以下の通りである．

- **Information Retrieval (IR)**: 代数モデルや確率モデルに基づいて文書間の語彙的類似度を計算する手法．この種の手法は通常，学習データを必要としない．（代表例*: VSM, LSI, BM25, Jensen-Shannon Divergence* 等）
- **Classical Machine Learning (ML)**: 人手による特徴量エンジニアリングを利用し，従来の機械学習アルゴリズムを用いてリンクの有無を分類する手法．（代表例*: SVM, Random Forest, Naïve Bayes, Logistic Regression* 等）
- **Deep Learning (DL)**: ニューラルネットワークを用いてテキストやコードの意味的特徴を自動的に学習・抽出する手法．事前学習済みモデルのファインチューニングを含む．（代表例*:*

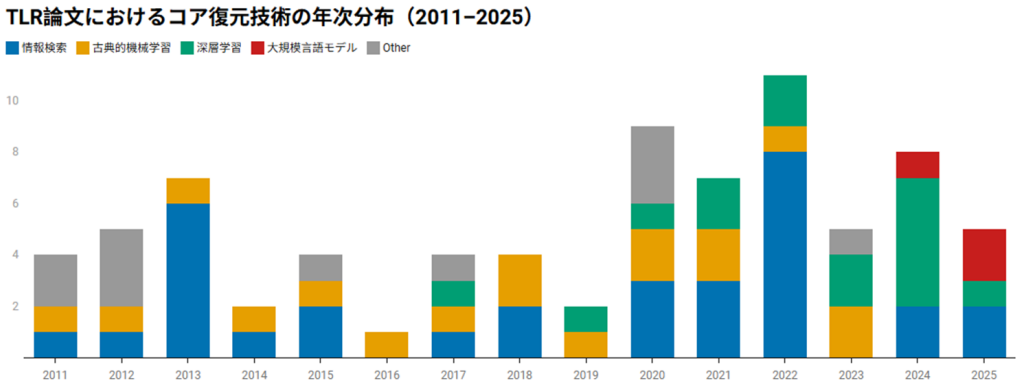


図 8　各リンク回復技術の論文数の年次推移（2011–2025）

*Word2Vec, CNN, RNN/LSTM, BERT/RoBERTa* 等）

- **Large Language Models (LLM)**: 数十億パラメータ以上の生成的大規模言語モデルに基づき，プロンプトエンジニアリングや文脈内学習を通じて直接推論を行う手法．（代表例*: GPT-3.5/4, LLaMA, Claude* 等）
- **Other**: 上記のいずれにも属さない手法．（代表例*: Static/Dynamic Slicing, Rule-based, Evolutionary Algorithms* 等）

特筆すべき点として，複数の技術を組み合わせたハイブリッド手法（例：ML を用いてクエリ拡張を行い，その後 IR で検索するなど）については，本研究では最終的にリンクのスコア判定やランキング決定を行う技術に基づいて分類した．例えば，前処理段階で機械学習技術を使用していても，最終的な類似度判定が VSM 等の情報検索モデルに基づいている場合，その本質的なアルゴリズム特性を鑑みて「IR」カテゴリに分類した．

### 4.3.2　技術トレンドの時系列的変化

図 8 は，2011 年から 2025 年における中核的リンク回復技術ごとの論文数の年次分布を示している．全体として，TLR 分野の技術パラダイムは単一的かつ線形な入れ替わりを示していない．すなわち，新技術の出現は旧技術を完全に淘汰したわけではなく，利用可能な技術手段を多様化させていると言える．

2011 年–2016 年において，情報検索技術は論文総数の過半数を占めている．従来の機械学習手法（黄色のバー）も継続して出現しているものの，情報検索技術が数量的に明らかな優位性を保っている．

2017 年から，深層学習技術に関する論文数が明確な増加傾向を示し始めた．特に 2020 年から 2022 年にかけて，深層学習技術の適用比率が著しく増加し，数量において従来の情報検索技術に徐々に接近し，一部の年では上回るようになった．

2023 年以降，グラフ上で最も顕著な変化は大規模言語モデルの出現である．2024 年および 2025 年

には，大規模言語モデル関連の論文数が爆発的な増加を示している．

注目すべき点として，深層学習や大規模言語モデル技術が大幅に増加した 2020 年代においても，情報検索や機械学習技術は完全に消滅しておらず，依然として毎年一定数の発表が維持されている．全体として，TLR 分野の技術的変遷は，多種多様な技術パラダイムが長期的に共存する形態をとっている．

#### 4.3.3 各技術アプローチの詳細

**Information Retrieval (IR)** 情報検索（IR）に基づく手法は，TLR 分野において最も広く適用され，かつ最も長い歴史を持つ技術パラダイムである．その基本原理は，ソフトウェア成果物（要求文書，ソースコードなど）を純粋なテキストと見なし，Bag-of-Words モデルを用いて高次元ベクトルに変換し，コサイン類似度や確率モデルを用いて成果物間の関連度を計算することにある．IR 手法の最大の利点はその教師なし（Unsupervised）特性にあり，事前にラベル付けされた学習データを必要としないため，実プロジェクトへの導入が極めて容易である [36]．しかし，異なる成果物（抽象的な要求と具体的なコードなど）が同じ機能を記述していても，全く異なる語彙が使用されるという，有名な「語彙の不一致（Vocabulary Mismatch）」問題にも直面している [45]．

この制約を克服するため，本システマティックレビューに含まれる IR 系の研究の多くは，ドメイン固有の知識や構造情報を導入することで，基礎的な IR モデルの性能向上に取り組んでいる．以下に，その例として 2 つの研究方向を紹介する．

- **De Lucia らによる研究（「ノイズ」に対するスムージング処理）[36]**: 従来の IR 手法（VSM や Jensen-Shannon Divergence など）は，成果物に含まれる「ノイズ語」の影響を受けやすい．De Lucia らは，スムージングフィルタ（Smoothing Filter）に基づく手法を提案した．この手法は，従来のストップワードリストのように単語を一律に削除するのではなく，コードの構造（制御フローグラフなど）や文書の文脈に応じて，単語の重みを動的に調整する．これにより，無関係な情報（Noise）による類似度計算への干渉を効果的に低減し，検索精度を向上させている．
- **Qusef らによる研究（プログラムスライシングを組み合わせたハイブリッド戦略）[79]**: Test-to-Code という特定のリンクタイプを扱う際，Qusef らは，テキストの類似度だけでは不十分であると指摘した．彼らは，LSI（潜在意味インデックス）と静的・動的プログラムスライシングを組み合わせた手法を提案した．LSI は同義語の問題（語彙の不一致の緩和）を処理するために用いられ，プログラムスライシングはコードの実行依存関係を利用して探索空間を縮小するために用いられる．この「IR + プログラム分析」というハイブリッド戦略は，特定のタスクにおける IR 技術の深化を典型的に表している．

**Classical Machine Learning (ML)**　テキストの類似度に主として依存する IR 手法とは異なり，古典的な機械学習（ML）手法は，リンク復元問題を「二値分類問題」として再定義する．すなわち，分類器（SVM, Random Forest, Naïve Bayes など）を訓練することで，ある成果物ペアの間に「リンクが存在する」か「存在しない)」かを判定する [14].

ML 手法の中核的な特徴は特徴量エンジニアリング（Feature Engineering）にある．ML 手法では，単なるテキスト特徴量（TF-IDF 値など）だけでなく，構造的特徴（クラス継承関係など）やメタデータ特徴（著者，タイムスタンプなど）といった多様な異種情報を結合して利用することが可能である．しかし，ML 手法は通常，教師あり学習（Supervised Learning）に属するため，その性能は高品質なラベル付きデータセット（Labeled Data）に強く依存する．したがって，十分な訓練データの確保や，データの不均衡（Data Imbalance）問題への対処が，この分野における重要な課題となっている [50].

以下に，その例として 2 つの研究を紹介する．

- **Mills による研究（検索から分類へのパラダイムシフト）[14]**: Mills は，従来の IR 手法が本質的に「検索」であり，ランキングリスト（Ranked List）を返すため，依然として開発者が人手で選別するために多大な労力を要すると指摘した．そこで彼は，機械学習分類器（Random Forest など）を直接用いてリンクの二値判定を行う手法を提案した．この研究は，問題を分類タスクとして定式化することで，人手による確認作業を効果的に削減でき，かつ閾値を設定した IR 手法よりも高い適合率（Precision）を達成できることを実証した．
- **van Oosten らによる研究（MDD に向けた特徴量エンジニアリング）[87]**: モデル駆動開発（MDD）の背景において，成果物（モデルファイルなど）は大量の非テキスト構造情報を含んでいる．van Oosten らは 131 種類の特徴量を含む分類器を開発した．これらの特徴量には，従来のテキスト類似度だけでなく，コミット（Commit）のサイズ，ファイルタイプ，パス類似度などのメタデータも含まれる．Gradient Boosting や Random Forest などのアルゴリズムを用いることで，この研究は，純粋なテキスト IR 手法では到達困難な，多次元特徴量の統合における ML 手法の強力な能力を示した．

**Deep Learning (DL)**　深層学習技術の導入は，従来の IR や ML 手法では克服が困難な「セマンティックギャップ（Semantic Gap)」問題を解決することを目的としている．Bag-of-Words (BoW) に依存する浅い手法とは異なり，DL モデルは多層ニューラルネットワークを通じて，ソフトウェア成果物の低次元密ベクトル表現（Dense Vector Representation/Embedding）を自動的に学習する．この表現空間では，意味的に類似しているが語彙が異なる用語（例えば「save」と「persist」）が近接した位置にマッピングされるため，モデルは文脈的意味（Contextual Semantics）を捉えることが可能になる．

さらに，DL 技術は「End-to-End 学習」のパラダイムを導入し，人手による特徴量エンジニアリングへの依存を減らし，ソフトウェア工学分野における事前学習済みモデル（Pre-trained Models）の適用を促進した [10].

以下に，代表的な 2 つの研究を紹介する．

- **Guo らによる研究（単語埋め込みと RNN の初期適用）[10]**: この分野で早期に深層学習を導入した研究の一つとして，Guo らは，VSM が語順や多義語を捉えられないという欠点に対処するため，単語埋め込み（Word Embedding）とリカレントニューラルネットワーク（RNN）を組み合わせたトレーサビリティネットワークを提案した．この手法は，領域固有の単語ベクトルを学習するために Word2Vec を利用し，要求文の時系列的意味をエンコードするために RNN を使用する．この研究は，従来の統計モデルと比較して，ニューラルネットワークに基づく意味表現が，深い意味的関連を持つリンクをより正確に識別できることを実証した．
- **Lin らによる研究（事前学習済みモデルの転移学習 - T-BERT）[54]**: BERT などの Transformer アーキテクチャの出現に伴い，Lin らは T-BERT フレームワークを提案し，TLR が「事前学習＋ファインチューニング」の時代に突入したことを示した．追跡データの希少性という問題に対処するため，T-BERT は段階的な学習戦略を採用している．まず，大規模な Wikipedia およびオープンソースコードデータで事前学習を行い，次に，関連するソフトウェア工学タスクデータを用いて適応学習を行い，最後にターゲットプロジェクトの追跡データでファインチューニング（Fine-tuning）を行う．この転移学習（Transfer Learning）メカニズムは，コードと自然言語に対するモデルの理解能力を大幅に向上させ，当時において BERT ベースの手法が State-of-the-Art の地位を確立する要因となった．

**Large Language Models (LLM)** 大規模言語モデル（LLM）の出現は，TLR 技術が新たな生成 AI 時代に突入したことを示している．「事前学習＋ファインチューニング（Fine-tuning）」のパラダイムに依存していた従来の DL モデル（BERT など）とは異なり，LLM に基づく手法は，主に「文脈内学習（In-context Learning）」と「プロンプトエンジニアリング（Prompt Engineering）」に依存している．LLM は膨大な一般知識と強力な推論能力を有しており，ゼロショット（Zero-shot）またはフューショット（Few-shot）の設定下で，複雑なソフトウェア成果物の意味を理解することが可能である [64]. これは，開発者がモデルを訓練するために大規模なラベル付きデータセットを構築する必要がなくなることを意味し，TLR 技術の実用化のハードルを大幅に下げるものである．現在の研究は主に，より効果的なプロンプトの設計方法や，LLM のハルシネーション（Hallucination）や入力長制限の問題を克服するための検索拡張生成（RAG）の活用に焦点を当てている [30].

以下に，現在関連する 3 つの研究を紹介する．

- **Hassine による研究（ゼロショットプロンプトエンジニアリングの検証）[64]**: Hassine は，汎用 LLM（GPT-3.5）を用いてゼロショット（Zero-Shot）設定下でリンクを回復する可能性を探索した．この研究は，セキュリティ要求（Security Requirements）とゴールモデル（Goal Models）という，抽象度の高い層間のリンクタスクに焦点を当てている．精巧に設計されたプロンプト（Prompt）を通じて，モデルに GRL（Goal-oriented Requirement Language）の意味を理解させるよう誘導した．実験の結果，LLM 自身の知識推論のみで，特定領域の訓練データを一切必要とせずに，100% の適合率（Precision）を達成できることが示された．これは，セマンティックギャップが極めて大きいタスクにおいて，LLM が驚異的な潜在能力を持つことを証明している．
- **Fuchß らによる研究（LiSSA: 検索拡張生成フレームワーク）[30]**: LLM のコンテキストウィンドウ制限（Context Window Limit）により，大規模プロジェクトの全コードを処理できないという問題に対し，Fuchß らは LiSSA フレームワークを提案した．この手法は検索拡張生成（RAG）メカニズムを導入している．まず軽量な IR 技術を用いてクエリに最も関連する候補成果物を検索し，次にそれらの候補を「コンテキスト」として LLM に与え，LLM に最終的な精密判定を行わせる．LiSSA は，汎用 LLM が特定領域のコードベースに直面した際の「知識不足」の問題を効果的に解決し，複数のデータセットにおいて，専用に微調整された BERT モデルを凌駕する性能を示した．
- **Wang らによる研究（MPLinker: 対抗的プロンプトチューニング）[59]**: 手動によるプロンプト設計は有効であるが，試行錯誤に依存しがちである．Wang らは Issue-Commit リンクタスクを対象に，自動化されたマルチテンプレートプロンプトチューニング（Multi-template Prompt Tuning）手法である MPLinker を提案した．モデル全体を微調整するのではなく，LLM のパラメータを凍結し，バックプロパゲーションを通じて「プロンプトテンプレート」自体のベクトル表現のみを最適化し，さらに対抗的学習（Adversarial Training）を導入してモデルの頑健性を高めた．この研究は，「人間がプロンプトを書く」から「アルゴリズムでプロンプトを最適化する」へと進化した，LLM 適用の高度な形態を示しており，特定タスクにおいて極めて高い性能を達成した．

## 4.4 RQ3 への回答

### 4.4.1 データセットの分類基準

本研究では，各論文で使用されたデータセットの入手可能性と性質に基づき，以下の 4 つのカテゴリに分類した．

- **公開データセットのみ**: 論文で使用したデータセットが，第三者でも入手可能な形で提示されて

いる場合を指す．例えば，データセット名や配布先（URL など）が明示されており，他の研究者が同一データを入手して再現実験を行うことが可能であると考えられるものを本カテゴリに含めた．

- **非公開・商用データセットのみ**: 企業や組織の内部データなど，第三者が一般には入手できないデータセットのみを使用している場合である．論文内で「公開不可」や「NDA（秘密保持契約）により共有不可」などの理由が示されているケースが多い．公開データが一切含まれない場合は本カテゴリとした．
- **公開＋非公開の併用**: 同一の論文内で，公開データセットと非公開（または商用）データセットの両方を組み合わせて評価を行っている場合である．
- **自作データセット**: 研究目的で人工的に作成された成果物を指す．これらは実世界に存在するプロジェクトではなく，学生への課題やトイ・プログラム（Toy Program）などが含まれる．

#### 4.4.2　データセットの分布状況

**全体的な分布**　全体的な割合を見ると，「公開データセットのみ」が最も主流な選択肢となっている．本システマティックレビューの対象論文 77 本における内訳は以下の通りである．

- **公開データセットのみ**: 55 本（約 71.4%）
- **非公開・商用データセットのみ**: 13 本（約 16.9%）
- **公開＋非公開の併用**: 6 本（約 7.8%）
- **自作データセット**: 3 本（約 3.9%）

この結果は RQ3 の核心的な結論を直接的に支持している．すなわち，TLR 研究は評価データにおいて，入手可能かつ再現性のある公開データセットに明らかに偏っており，企業や実際の現場データを用いた評価研究は相対的に不足していると言える．

**年次分布**　図 9 の年次ごとの積み上げ棒グラフから，以下の 2 つの事実が読み取れる．

1. **近年（特に 2020 年以降）の論文数の増加**: 全体的な研究熱が高まっており，関連研究が近年に集中している．
2. **公開データセットの優位性**: 研究数が増加しているにもかかわらず，構成比としては依然として「公開データセットのみ」が主体である．多くの年において，棒グラフの主要な高さは公開データセットによって占められており，これが TLR 研究のデフォルトの選択肢であり続けていることを示している．

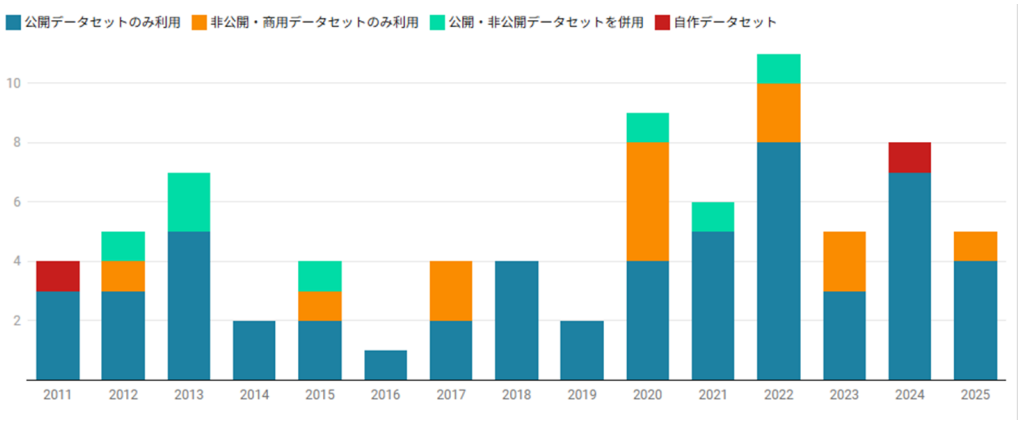


図 9　データセットタイプの年次分布推移

同時に，以下の詳細な変化も観察できる．

- 「非公開・商用データセットのみ」および「公開＋非公開の併用」は，完全に不在ではないものの，通常は各年の研究の一部を占めるに留まっている．
- 「自作データセット」は極めて稀であり，時間軸上でも散発的な分布を示しており，継続的な利用増加の傾向は見られない．

本レビューが対象とした TLR 論文における評価データセットの選択は，全体として公開データセット主導の分布特徴を示している．非公開・商用データセットのみの使用や，公開・非公開の混合評価も存在するが，その比率は小さい．自作データセットは最も稀である．全体として，研究評価は再現可能な公開データに集中しており，企業の現場データに基づく体系的な評価は依然として不十分である．

### 4.5　RQ4 への回答

RQ4 に回答するため，本節では RQ1 から RQ3 の統計結果に基づきクロス集計分析を行う．具体的には，表 5（RQ1 × RQ2）と表 6（RQ1 × RQ3）を構築し，それぞれ異なるリンクタイプと技術アプローチ，およびデータセットの公開性との間の共起関係を描写する．以下に，これら 2 つのクロス集計表から得られた主要な観察結果を順次報告する．

#### 4.5.1　クロス分析の目的と集計基準

RQ1 から RQ3 では，それぞれリンクタイプ（link types），技術アプローチ（approaches），データセットの公開性（dataset openness）の 3 つの次元から，既存の TLR 研究の全体像を描写した．しかし，これら 3 つの次元は実際の研究において相互に独立しているわけではない．異なる成果物間のリンクタスクは，しばしば異なる意味的粒度，構造情報，入手可能なデータソースに対応しており，それら

が技術選択や再現性に制約を与える可能性があるためである．

RQ4「成果物，技術，データセットの間にどのような関係性と課題が存在し，それに基づいてどのような将来の方向性が示唆されるか」に答えるため，本研究ではさらにクロス分析を実施した．RQ1 と RQ2，RQ3 の結果を結合して統計することで，リンクタイプ・技術アプローチ・データセット公開性の間の共起構造と潜在的な関係を明らかにする．

具体的には，本節では以下の 2 つのクロス集計表を構築する．

- **表 5（RQ1 × RQ2）**：リンクタイプを行，技術カテゴリを列とし，各リンクタイプにおいて IR / ML / DL / LLM / Other 等の技術アプローチが出現した回数を集計する．
- **表 6（RQ1 × RQ3）**：リンクタイプを行，データセットの公開性カテゴリを列とし，各リンクタイプにおいて「公開データセット」「非公開・商用データセット」「公開＋非公開の併用」「自作データセット」等のデータソースが出現した回数を集計する．

表 5　リンクタイプと技術アプローチのクロス集計（RQ1 × RQ2）

| リンクタイプ | IR | ML | DL | LLM | Other |
|---|---|---|---|---|---|
| Requirement – Code | 18 | 6 | 1 | 1 | 4 |
| Issue – Code | 2 | 5 | 10 | 1 | 2 |
| Requirement – Requirement | 5 | 5 | 0 | 0 | 2 |
| Requirement – Design | 5 | 6 | 1 | 1 | 2 |
| Design – Code | 5 | 4 | 0 | 0 | 2 |
| Code – Test | 6 | 3 | 1 | 0 | 0 |
| Requirement – Test | 5 | 4 | 0 | 0 | 0 |
| Design – Test | 2 | 3 | 0 | 0 | 1 |
| Code – Document | 1 | 0 | 0 | 1 | 2 |
| Design – Document | 0 | 0 | 0 | 1 | 1 |
| Test – Document | 1 | 0 | 0 | 0 | 0 |
| Issue – Issue | 1 | 0 | 1 | 0 | 0 |
| Requirement – Document | 0 | 0 | 0 | 0 | 1 |
| Issue – Design | 0 | 1 | 0 | 0 | 0 |
| Issue – Test | 1 | 0 | 1 | 0 | 0 |

後続の小節では，以下の問題に重点を置いて議論を行う．

1. 異なるリンクタイプが，安定した技術アプローチの選好を示しているか．

表 6　リンクタイプとデータセット公開性のクロス集計（RQ1 × RQ3）

| リンクタイプ | 公開のみ | 非公開のみ | 併用 | 自作 |
|---|---|---|---|---|
| Requirement – Code | 19 | 5 | 5 | 1 |
| Issue – Code | 18 | 1 | 1 | 0 |
| Requirement – Requirement | 10 | 1 | 1 | 0 |
| Requirement – Design | 7 | 5 | 2 | 1 |
| Design – Code | 8 | 1 | 2 | 0 |
| Code – Test | 8 | 0 | 2 | 0 |
| Requirement – Test | 6 | 0 | 3 | 0 |
| Design – Test | 5 | 1 | 0 | 0 |
| Code – Document | 2 | 1 | 0 | 0 |
| Design – Document | 1 | 1 | 0 | 0 |
| Test – Document | 1 | 0 | 0 | 0 |
| Issue – Issue | 1 | 1 | 0 | 0 |
| Requirement – Document | 0 | 1 | 0 | 0 |
| Issue – Design | 0 | 1 | 0 | 0 |
| Issue – Test | 0 | 1 | 0 | 0 |

2. データセットの公開性が，特定のリンクタイプと有意に関連しているか．

#### 4.5.2　リンクタイプと技術アプローチの対応関係

表 5 は，異なるリンクタイプと技術アプローチ（IR / ML / DL / LLM / Other）のクロス分布をまとめたものである．全体として，既存の研究は「リンクタイプ—手法路線」の組み合わせにおいて，比較的明確な層別化構造を示している．一方では，研究量の多い主流リンクタイプにおいてすでに相対的に安定した主導的な技術路線が形成されている．他方で，サンプル数の少ないロングテールなリンクタイプでは，一貫した技術パラダイムが未形成であり，手法の選択がより分散している．

まず，研究量が最大である 2 つのリンクタイプにおいて，技術路線は明確な分化を示している．Requirement–Code については，IR が主導的な地位（18 件）を占め，一定規模の ML 手法（6 件）がこれを補完している．これに対し，DL と LLM の出現は依然として少なく（各 1 件），同時に少数の「Other」類の手法（4 件）が存在し，このリンクタイプが依然としていくつかの非典型的または特定シナリオ向けのアプローチを含んでいることを反映している．これと対照的なのが Issue–Code である．このタイプでは DL が顕著に優勢（10 件）であり，ML も一定の割合（5 件）を占めるが，IR の出現頻度は低い（2 件）．この結果は，同じ「実装側をターゲットとする」リンクタスクであっても，異なるリンクタイプ間では，既存研究においてすでに異なる主流技術の系譜が形成されていることを示している．

次に，要求や設計など上流成果物間のリンクタイプにおいては，IR と ML の使用がより均衡してお

り，DL/LLM の採用は相対的に限定的である．例えば，Requirement–Requirement では IR と ML が近い出現回数（5 対 5）を示し，Requirement–Design も IR と ML（5 件と 6 件）をカバーしつつ，DL と LLM は散発的な出現（各 1 件）に留まっている．Design–Code も主に IR と ML（5 件と 4 件）で構成されており，DL/LLM は目立った規模を形成していない．総じて，上流リンクタイプの研究は伝統的な検索および学習手法を中心に展開されており，深層学習や生成 AI モデルが主流リンクと同等の規模で適用される傾向はまだ観察されていない．

最後に，文書関連やクロスドメインなロングテールリンクタイプでは，手法選択において「低頻度かつ多様」という特徴が見られる．Code–Document を例にとると，IR，LLM および Other がいずれも出現しているが（それぞれ 1，1，2 件），全体数は限られている．Design–Document でも LLM と Other が出現（各 1 件）しているが，IR/ML/DL はほぼ不在である．同様に，Issue–Design や Issue–Test などのタイプも全体的なサンプル数が少なく，安定した技術的主流を形成するには至っていない．この現象は，ロングテールなリンクタイプにおいては研究が尚も探索段階にあり，手法体系の再利用性や比較可能性についてさらなる蓄積が必要であることを示唆している．

### 4.5.3 リンクタイプとデータセット公開性の関係（RQ1 × RQ3）

表 6 は，異なるリンクタイプにおけるデータセットの公開性と入手源に関するクロス分布をまとめたものである．これには「公開データセットのみ利用」「非公開・商用データセットのみ利用」「公開・非公開データセットを併用」「自作データセット」のカテゴリが含まれる．全体として，データセットの公開性は異なるリンクタイプ間で明らかな差異を示している．主流なリンクタイプは概して高い公開データの可用性を持っているのに対し，若干のロングテールなリンクタイプは非公開データにより強く依存しており，その結果，再現性や研究間の比較において構造的な制限を受けている．

まず，研究量が最大である 2 つのリンクタイプについて見ると，いずれも公開データセットが主導的であるが，その構造は同一ではない．Requirement–Code は「公開データセットのみ」が主体（19 件）であるが，同時に一定規模の「非公開・商用データセットのみ」（5 件）および「併用」（5 件）が存在し，少数の「自作データセット」（1 件）も見られる．これと比較して，Issue–Code はサンプルにおいてほぼ完全に公開データセットによって支えられている．「公開データセットのみ」のカウントは 18 に達する一方，非公開や併用はいずれもわずか 1 件であり，自作データセットは 0 件である．この結果は，同じ主流リンクタイプであっても，異なる研究方向ではデータソースの構造に差異があり，前者はより多元的な混合形態を示しているのに対し，後者は公開データへの依存がより集中的であることを示している．

次に，要求，設計，テストなどの上流成果物間のリンクタイプにおいては，依然として公開データセットが主導的であるが，一部のタイプでは非公開データの割合がより顕著である．例えば，Requirement–

Design の公開データセットのカウントは 7 であるが，同時に「非公開・商用のみ」が 5 件,「併用」が 2 件,「自作」が 1 件あり，この種のリンクがデータの入手性においてより明確な多源併存の特徴を持っていることを示している．これと対照的に，Requirement–Test や Design–Test などのタイプは公開データが主（それぞれ 6 件と 5 件）であり，非公開や併用の割合は低いか，散発的な分布に留まっている．総じて，上流リンクタイプの研究は公開データを基盤に展開可能ではあるものの，リンクタイプによってデータソースの分化には濃淡が存在する．

さらに重要な点として，若干のロングテールなリンクタイプは，サンプルにおいて公開データセットによる支えがほぼ欠如している．例えば，Requirement–Document，Issue–Design，Issue–Test では,「公開データセットのみ」の項目がいずれも 0 件である一方，いずれも「非公開・商用のみ」が出現（各 1 件）しており，公開・非公開の併用や自作データセットによる補完も観測されなかった．同様に，Design–Document や Code–Document などの文書関連リンクタイプは公開データが存在するものの，全体規模は小さい（例えば Code–Document の公開データは 2 件）．この現象は，ロングテールなリンクタイプにおいて，公開ベンチマークデータの欠如が研究の蓄積と手法の比較可能性を制限する重要な構造的要因であることを示唆している．

以上をまとめると，表 6 はリンクタイプとデータセットの公開性との間に顕著な共起構造が存在することを示している．すなわち，主流リンクタイプは全体としてより強力な公開データの可用性を備えているのに対し，ロングテールなリンクタイプは非公開データソースにより依存している．

以上のクロス集計により，リンクタイプ，技術アプローチ，データセットの入手可能性の間の共起構造が明らかになった．これらの構造的差異の原因，それによってもたらされる研究上の課題，および今後の方向性については，第 5 章でさらに詳しく議論する．

# 5　考察

## 5.1　RQ1–RQ4 回答結果の総括

本研究では，2011 年から 2025 年の間に選定された 77 本の TLR（Traceability Link Recovery）関連論文を対象に，リンク種別（RQ1），技術アプローチとその進化（RQ2），データセットの公開性（RQ3），およびこれら 3 者の共起構造と偏り（RQ4）について統一的なコーディングと統計分析を行った（第 4 章の各図表を参照）．これらの結果は，後続の考察のための事実基盤を提供するものである．

**RQ1：リンク種別分布は「主流への集中とロングテールの分散」を示す**　成果物タイプ（Requirement / Design / Code / Test / Issue / Document）により構成されるリンクマトリクスにおいて，研究は Requirement–Code（30 本）と Issue–Code（20 本）の 2 種類に著しく集中しており，その他のリンク種別は明らかに少なく，ロングテールを形成している．同時に，多数の論文が単一のリンク種別のみを扱っており，複数のリンクタスクを同時に処理する論文は少数である（図 2 参照）．これは，本分野の主流研究が往々にして単一かつ明確な工学的課題（Pain Points）に焦点を当てていることを示唆している．方向性の統計はさらに，リンク回復タスクが全体として上流から下流への検索・特定パラダイム（例：Requirement→Code，Issue→Code の方向設定が主導的）に偏っており，逆方向や無向・双方向の設定は相対的に少ないことを示している（表 4 参照）．

**RQ2：技術の系譜は線形的な代替ではなく，長期的共存と 2020 年以降の加速的な分化を示す**　IR / ML / DL / LLM / Other の 5 技術カテゴリによるコーディングと年次推移（図 5 参照）は以下の傾向を示した．2011 年から 2016 年にかけては IR が長期にわたり数的優位を占めた．2017 年より DL が明確な成長を開始し，特に 2020 年から 2022 年にかけて急速に IR に接近し，一部の年では上回った．2023 年以降は LLM が新技術として登場し，2024 年から 2025 年にかけて顕著な増加傾向を示している．同時に，IR と従来の ML は新パラダイムの出現によって淘汰されることなく，複数の年で一定数を維持しており，新旧技術の共存と適用シーンの分化という進化形態を体現している．

**RQ3：データセット利用は「公開データセットのみ」が主導的であり，全期間を通じて安定的である**　データの入手可能性に関しては，全体として「公開データセットのみ（55 本）」が圧倒的多数を占めている．対して，「非公開（産業/商用）のみ（13 本）」，「公開・非公開の併用（6 本）」，および「自作データセット（3 本）」の割合は低い．年次分布（図 6 参照）によると，2020 年代に入り研究総数が増加しても，「公開のみ」の主導的地位は大きく変化していない．これは，本分野の再現可能な実証体系が，入手可能な公開ベンチマークとオープンソースエコシステムに大きく依存していることを意味する．

**RQ4：リンク種別・技術・データ公開性の間に三次元横断的な構造が存在する**　クロス集計は，2 つの重要な共起関係を明らかにした（表 5，表 6 参照）．第一に，主流リンク種別において相対的に安定した技術的選好が形成されている．例えば Requirement–Code は IR を主体（18 本）とし ML が補完する（6 本）のに対し，Issue–Code は DL が顕著に主導（10 本）しており IR は少ない（2 本）．第二に，データ公開性においてリンク種別ごとに明確な差異が見られる．Issue–Code は公開データへの依存が極めて強い（18/20）．Requirement–Code は公開主体（19/30）であるが，非公開や併用の割合が相対的に高い（5+5）．一方で Requirement–Design の「公開のみ」の比率は低く（7/15），非公開や併用・自作がより一般的である（5+2+1）．

以上の第 4 章の統計結果は，明確な傾向が確認できる．すなわち，研究タスクは少数の主流リンク種別に集中し，技術路線は 2020 年以降に顕著に分化したものの代替は起きておらず，再現可能な実証は公開データに高度に依存している．そして，主流とロングテールの間で技術とデータ条件に構造的な成層化（Stratification）が生じている．これらの事実に基づき，本章の以降の節では，なぜ RQ1 において主流集中とロングテール化が生じるのか（5.2 節），なぜ IR が長期安定し学習ベース手法が近年加速しているのか（5.3 節），なぜ公開データセットが長期優位にあるのか（5.4 節），そしてなぜリンク種別・技術・データ公開性の間に共起構造が出現するのか（5.5 節）について，それぞれ詳細に考察する．さらに，これらの分析結果を踏まえ，研究コミュニティおよび実務家に対する具体的な提言を行い（5.6 節），最後に本研究の限界と妥当性への脅威について総括する（5.7 節）．

## 5.2　リンク種別分布の解釈（RQ1）

### 5.2.1　Requirement–Code と Issue–Code に研究が集中する理由

RQ1 の統計結果が示す通り，Requirement–Code および Issue–Code は他のリンク種別と比較して圧倒的な割合を占めている．以下に，この 2 種類のリンクがなぜ TLR 研究において支配的であるのか，その背景と要因を分析する．

**(1) Requirement–Code**　Requirement–Code（要求–コード）リンクが TLR 研究において最も高い割合を占める理由は，以下の 4 点に集約される．

**第一に，ソフトウェア工学活動における基礎的な価値の高さである．**複数の研究が指摘するように，要求追跡は「ソースコードが要求と一致しており，かつ規定された要求のみを実装していること」を保証するための重要な手段である．保守や進化の段階において，追跡情報は理解コストを低減し，影響分析などの活動を支援する [45, 37]．さらに，規制産業やセーフティクリティカルなシナリオでは，追跡はコンプライアンス検証やカバレッジ分析に不可欠であり，認証や承認の要件と直接結びついている．

そのため，Requirement–Code は他のリンク種別以上に，学界と産業界の双方が解決すべき「収益の明確な核心的課題」として認識されやすい．

**第二に，実践におけるリンクの形骸化とそれに伴う回復需要の高さである．**一方では，保守や進化の過程において，開発者が更新コストを継続的に負担できず，追跡リンクが陳腐化しやすいことが指摘されている．そして，事後的にこれらのリンクを回復することは困難かつ高コストである [45]．他方で，プロセス的な視点からは，プロジェクトにおける「追跡情報の明示的な記録」は開発者にとって負担が大きく，記録の不備や情報の陳腐化を招きやすい．これにより，自動または半自動による追跡生成・回復は魅力的な代替手段となるが，同時に大量の候補関係をスクリーニングしなければならないという新たな問題も生じる [34]．この「高い価値」と「高い保守コスト」の組み合わせが，Requirement–Code を常に主要な研究対象へと押し上げている．

**第三に，TLR 分野で長らく主流である IR（情報検索）型モデリングおよび評価パラダイムとの整合性である．**多くの研究において，追跡回復は「要求（またはユースケース等の上位テキスト）をクエリとし，コードファイル等を文書集合とみなして類似度を計算し，候補ペアをランク付けする」タスクとして記述される [36]．また，手動による追跡維持の人的コストが半自動回復技術への関心を高め，長期間にわたり IR がこのプロセスを部分的に自動化するために採用されてきた [36]．この基盤の上で，いかに開発者に候補リストを効率的に分析させ，誤判定やスクリーニングコストを削減するかという改善も重要な研究方向となっている．この「候補ランキングリスト＋人間による確認」という評価モデルは，Requirement–Code のデータ形態と高度に適合しており，再現可能な実験設定と方法論の進化を形成しやすい．

**第四に，最も典型的かつ顕著な「意味論的ギャップと語彙の不一致」のシナリオであり，アルゴリズムの試金石となっている点である．**抽象度の異なる成果物（要求とコード）は，同一の概念を異なる用語で表現する傾向があり，これが語彙の不一致を引き起こし，テキスト類似度に基づく IR の効果を著しく低下させる．同時に，このシナリオでの手動回復は，概念レベルの差異と回復すべきリンクの数により労働集約的でエラーが発生しやすい．これは自動化研究の必要性をさらに強化する論拠となる [15]．総じて，Requirement–Code は「重要であるが困難である」という特徴を併せ持っており，追跡技術の核心的な課題を反映すると同時に，新手法にとって最も代表的な検証シナリオを提供するため，文献分布上の集中をもたらしている．

**(2) Issue–Code**　Requirement–Code と同様に，Issue–Code が集中して出現する背景には，以下の明確な理由が存在する．

**第一の理由は，保守価値の明確さと直接性である．**Issue とそれを修正したコード変更をリンクすることは，保守作業を支援し，「保守コストを大幅に削減できる」と指摘されている [61]．また，これらのリンクは保守関連活動（Issue と PR/commit 等の開発活動の関連付けなど）を支援する基盤となる [63]．

**第二の，より核心的な理由は，実プロジェクトにおける「リンクの欠落」と，それによる強力な回復・生成の動機である．**実際の開発リポジトリにおいて，この種のリンクは本来重要であるにもかかわらず，頻繁に欠落または不完全な状態にある．複数の論文が以下のように指摘している．

- 追跡リンクは保守に有益な情報を記録しているが，頻繁に欠落している [58].
- Issue–Commit リンクは良き実践とされるが，開発圧力の下では無視されがちであり，実際にリンクされている Issue は一部にとどまるという実証データがある [56].
- Bug–Change シナリオにおいて，開発者はバグ参照（Bug References）を書き忘れたり，不規則な記述を行ったりするため，単純なヒューリスティクスに基づくリンクは信頼性に欠ける [48, 49].
- PR–Issue などの広義の協力成果物間のリンクも頻繁に省略される [62].
- 手動によるリンク作成・ラベル付け自体も，時間がかかりエラーが発生しやすい [57, 59].

**第三の理由は，データセットの可用性と特性における優位性である．**Design や Test 等の成果物と比較して，Issue，Commit，PR 等は GitHub や Jira 等のプラットフォーム上で大規模かつ履歴・文脈付きで利用可能である．また，これらは「明示的な引用」や「Close 関係」などを通じて，比較可能な「正解データ（Ground Truth）」または「弱正解データ」を形成しやすい．この高密度な観測可能データは，大規模評価，半教師あり/弱教師あり学習，およびグラフ学習などの複雑な手法を適用するのに特に適している．

**結論として**，Requirement–Code は要求の実現と検証における核心的なリンクでありながら長期的な保守コストと欠落の問題に直面しており，一方で Issue–Code はオープンソース開発基盤におけるデータの可用性と実践上のリンク欠落という課題を背景に持っている．これら二つの要因が，これら 2 種類のリンクを TLR 研究の主役に押し上げていると言える．

### 5.2.2 他リンク種別が少数である理由

Requirement–Code や Issue–Code と比較して，その他のリンク種別が研究において少数にとどまる背景には，以下の 3 つの核心的な要因が存在する．

**(1) データ取得と検証の障壁**　第一の核心的な理由は，一部のリンクが実工学プロセスにおいて安定的かつ明示的に記録されておらず，研究者が高い「データ取得および検証コスト」を負担しなければならない点にある．例えば，脆弱性レポートに向けたファイルレベル追跡の研究では，「CVE–脆弱性関連ファイル」のペアデータセットが以前は存在しなかったことが指摘されている．そのため，研究者は CVE–パッチコミットから出発し，ヒューリスティクスを用いて必要なファイルレベルの正解データを逆推定（推論）する必要があった．さらに，セキュリティデータベースは「脆弱性関連ファイル」を明示的に提供しないため，正解データの構築は一層困難となる [55]．このように「手法を開発する前に，ま

ずデータを作成しなければならない」という高い参入障壁は，必然的に当該方向の論文数を抑制する要因となる．

**(2) 暗黙知への依存と汎用化の困難さ**　第二に，一部のリンクの成立は領域知識や「暗黙知（tacit knowledge）」に強く依存しており，その結果，転移可能な汎用手法の構築や，再現・比較実験が困難になるという点が挙げられる．NLP を活用したソフトウェア追跡のレビューでは，ソフトウェアアーキテクチャに関連する追跡において，暗黙知がリンク回復に「不可欠（vital）」であることが指摘されている．同時に，追跡対象は工学プロセスの中で絶えず進化し，成果物間の表現の差異も不確実性を増幅させる [89]．これは，多くの「非主流リンク」の研究が，大規模かつ蓄積可能な汎用ベンチマークを形成しにくく，特定のシナリオやドメインに特化した研究になりがちであることを意味している．

**(3) 高コストによるデータ供給の圧縮**　最後に，工学的適用の観点から見ると，手動による作成・保守コストの高さが，逆説的に「研究可能なデータ」の供給不足を招いている．ドキュメント–コード間の追跡に関する研究は，手動による追跡リンクの作成と保守が「時間を要しエラーが発生しやすい（time-consuming and error-prone）」ものであると指摘しており，その原因を成果物の抽象レベルの差異による意味論的ギャップに帰している．また，既存の自動化追跡研究は往々にして「要求とソースコード」のみをカバーしており，他の成果物タイプ（アーキテクチャ文書など）は重要でありながら対象とされにくい現状も言及されている [75]

### 5.3　技術選択と進化の背景（RQ2）

#### 5.3.1　なぜ IR は継続的に使用され続けているのか？

**(1) 自動化への直接的な入り口としての IR**　研究は，トレーサビリティが影響分析やコンプライアンス検証などの活動を支援し，FAA や FDA などの認証・承認の文脈において必須または重要な地位にあることを明確に指摘している．一方で，システム規模と複雑さの増大に伴い，必要なリンク数が急増するため，手作業による作成と保守は「抑制的（inhibitive）」になる．そのため，コミュニティは長期にわたり自動化手法に投資しており，IR はその直接的な可用性から，最も早く，かつ最も安定した技術的主流の一つとなっている [69]．

**(2) 人間参加型ワークフローとの親和性**　多くの研究が同一の基本ワークフローを踏襲している．すなわち，IR はテキスト類似度に基づいて候補リンクをランク付けし，上位 N 件のみを開発者・分析者の確認用に返す．これにより，IR は実際のプロセス（完全自動化ではなく，人間が確認すべき範囲の大幅な削減を目指すもの）に容易に組み込むことができる．しかし同様に，IR の精度は往々にして低く，自動化によって明らかに誤ったリンクを大量に排除できたとしても，分析者は依然として最終判定にかな

りの時間を費やす必要があることが指摘されている [73, 69].

**(3) 言語現象による精度の限界と改良への動機**　IR の核心的な弱点は致命的で利用不可能というわけではなく，言語現象の影響を受けやすい点にある．例えば，同一用語が異なる要求で異なる意味を持つ場合（多義性），IR ベースの追跡回復の精度は著しく低下する．研究はこの点について「IR（情報検索）ベースのトレーサビリティ回復は，多義性（polysemy）の影響により精度が低下する」と直接言及し，多義語を識別して精度を向上させるなどの改善策を提案している [73]．さらに，より「異質な言語・文法」を持つ成果物ペア（例：形式仕様 vs Java 実装）において，IR ベースの手法は過剰な偽陽性（False Positives）を返すことが指摘されており，その主要因の一つは知識の断絶や用語の不一致（knowledge gap / term mismatch）にある．したがって，研究は IR を完全に放棄するのではなく，「IR フレームワーク内に追加の証拠や特徴量を導入する」方向へと進む傾向がある [41].

**(4) データとコストの制約による現実解**　高精度が求められる追跡タスクにおいて，ML/DL 手法の適用を試みる際，大規模かつ網羅的な訓練データの欠如という問題に直面し，実行可能で高精度な結果を得ることが困難であることが明確に述べられている．これにより，大規模な注釈付きデータを必要とせずに候補とランキングを出力できる IR（または拡張 IR）の方が，工学的に実現可能性が高いとされる [41]．同様に，深層学習による追跡研究においても，DL は訓練データのスパース性や産業レベルでの適用コスト（訓練・推論が相対的に遅い）に制約されるため，当面の間は IR や ML がより一般的で配備可能な選択肢であり続けると指摘されている [54].

したがって，新技術が登場しても旧来手法が完全に置き換わるのではなく，IR は入り口として残り続け，必要に応じて学習ベース手法と併存・補完関係を形成する傾向がある．

### 5.3.2　学習ベース手法（ML/DL/LLM）台頭の背景：意味理解能力から実プロジェクトでの可用性へ

第一に，TLR における IR の長期的ボトルネックの一つは，成果物テキストの背後にある意味やドメイン知識を理解することが難しく，「一見関連しているが実際は誤り」な候補リンクを提示しやすい点にある．Guo らは，既存の自動化手法は IR を使用しているものの，「意味理解やドメイン知識の融合が困難」であり，その結果「不正確・不精密」な結果を生みやすいと指摘し，これを深層学習による意味表現導入の動機としている [10]．同様に，Zhu らも問題提起の中で，既存手法が Issue と Commit の意味的関係を理解しきれずリンク回復精度に影響している点を強調し，より強力な意味的関連性を捉えるために深層学習へと転換した [58].

第二に，学習ベース手法の台頭は，より工学的な現実にも起因している．実プロジェクトにおいてリンクは往々にして不完全かつ信頼性に欠け，追跡ネットワークが真に機能しにくいという問題である．

例えば，Rath らによる 6 つのオープンソースプロジェクトの分析では，平均して約 60% のコミットしか具体的な Issue にリンクされておらず，これらの基礎リンクの欠落がプロジェクトレベルの追跡集合を不完全にし，信頼に値しないものにしている [53]．同時に，Lin らも「手動作成・保守の高コスト」と「現実におけるリンクの不完全・不正確さ」を常態的な問題として述べ，これが追跡データが開発者に信頼されず，低頻度でしか利用されない原因であると指摘している [54]．そのため，研究界は単にランキング精度を向上させるだけでなく，学習ベースの手法を用いて欠落リンクを補完し，可用性を高めること（例えば分類器を用いてコミットメッセージ中の欠落した Issue タグを識別し，リンクを生成するなど）も試みている [53]．

手法レベルでは，深層学習や事前学習済み言語モデルの導入は，より強力な意味表現を獲得することを核心的な目的としており，いくつかの研究において IR に対する顕著な性能向上が報告されている．例えば，Guo らの深層学習追跡ネットワークは，VSM や LSI などの古典的 IR 手法よりも「有意に優れている」と報告されている [10]．Lin らも，事前学習済み言語モデルと転移学習によってデータ不足を緩和した後，そのフレームワークが評価プロジェクトにおいて古典的 IR 手法を凌駕し，VSM に対する MAP の大幅な向上を示したことを強調している [54]．このような「IR と直接比較して優位性を示す」結果が，DL/PLM をその後の主要な研究路線の一つへと押し上げた．

しかし，TLR における学習ベース手法への移行は無償のアップグレードではなく，その普及は新たな問題と対策を生み出した．最も典型的な課題は教師データの不足である．Lin らは要旨において，深層学習追跡モデルの有効性を制限する要因の一つとして「ラベル付きデータの可用性（labeled data availability）」と実行効率を挙げている [54]．Zhu らはより明確に，DL は従来の IR より優れているものの，往々にして「十分なラベル付きデータに高度に依存」しており，手動によるリンク注釈は時間と労力を要し，ドメイン専門知識も必要であるため，現実のプロジェクトでは「少量のラベル付きデータ＋大量のラベルなしデータ」という構成が一般的であると指摘している．さらに，Zhu らは大規模な注釈付きデータセットの構築は通常非現実的であるため，「既存のラベルなしデータの活用」が有望な方向性であると述べている [57]．より実データの分布に近い視点から，Dong らはさらに，学習ベースの追跡がデータの不均衡やスパース性といった重要な課題に直面していると総括し，半教師あり学習や擬似ラベルなどのメカニズムを用いてこれを緩和している [50]．これら一連の研究は，2010 年代以降に「ML/DL ＋ 半教師あり/弱教師あり/ラベルなしデータ利用」の組み合わせが大量に出現した理由，すなわち「DL はラベルに依存するが，現実はラベル獲得が困難である」という工学的ジレンマへの直接的な回答であることを説明している．

最後に，LLM 路線（およびプロンプト/検索拡張）が近年注目されている重要な理由は，「成果物ペアごとに個別に訓練・チューニングが必要」という断片化されたパラダイムから脱却し，より汎用的な TLR へと向かおうとしている点にある．Fuchß らは，既存の手法は往々にして「タスク特化型

(task-specialized)」であり，異なる追跡タスクのために個別に設計・訓練する必要があると指摘している．対して LLM は，多様なタスクに対する潜在的な「広範な適用可能性」を備えているが，コンテキストウィンドウや推論コストの制約があるため，関連情報を検索（RAG）してから LLM に推論させる必要があると述べている [30]．これに呼応するように，Wang らは Issue–Commit タスクをプロンプト学習として定式化し，マルチテンプレート・プロンプトチューニングを提案して効果を向上させている [59]．これは，研究の動機が単なる「類似度ランキングの向上」から，「タスク移行コストの削減およびクロスシナリオでの再利用性の向上」へと拡大していることを証明している．

### 5.4　データセットの偏り（RQ3）

#### 5.4.1　なぜ全期間を通じて「公開データセットのみ利用」が支配的であるのか？

本研究の統計結果が示す通り，2011 年から 2025 年の TLR 研究において，「公開データセットのみ利用」が継続的に支配的な地位を占めている．この現象は単なる研究者の選好ではなく，入手可能性の制約，再現性の規範，およびアノテーションコストの構造によって複合的にもたらされた結果であると考えられる．

**(1) 産業データの入手制約：成果物の専有性と機密性**　TLR に必要な主要成果物は，企業環境において営業秘密，顧客情報，またはコンプライアンス要件と結びついていることが多く，外部共有に対する天然の障壁が存在する．Gotel らは，トレーサビリティのロードマップ議論において，研究と実践の障壁の一つとして「専有的かつ機密性の高いデータ（proprietary and sensitive)」に直面することを明確に指摘し，さらにそのような状況下で組織が「教訓を共有することに消極的（reluctance to share lessons learned)」であると述べている [13]．このような制約は，利用可能な研究データを OSS（オープンソースソフトウェア）エコシステムへと直接的に誘導する．オープンソースプロジェクトでは，コードリポジトリ，コミット履歴，Issue/PR 議論が本質的に公開されているため，データ入手の障壁が低い．要するに，「公開データセットのみ利用」の支配は，入手可能性の制約だけでなく，再現性・比較可能性を最優先する学術規範が研究設計を公開ベンチマークへと誘導することによって強化されている．

**(2)「再現可能・比較可能」な学術規範による公開データ利用の促進**　TLR 手法は実験的な比較（ベースラインとの比較，同種手法との比較，論文間の対照）に高度に依存している．再現可能な実験と比較可能な評価を実現するため，研究コミュニティは長期にわたりベンチマーク化とオープン共有を提唱してきた．Gotel らは「技術を相対的に評価するためのベンチマーク基準（benchmarking standards for comparatively evaluating techniques)」の必要性を提起し，より大規模な実験評価を支援するために「産業データセットの共有リポジトリ（shared repository of industrial datasets)」の構築を呼びかけている [13]．同様に，Huang らは研究論文において，「公開されたデータセット（publicly available

datasets)」を使用し，「既存ベンチマークに対する（against previous benchmarks)」対照評価を行うべきであると強調している [12]．個別の研究論文においても，この公開・再現性の規範は具体的に反映されている．例えば Mazrae らは Issue–Commit リンク回復研究において「ソースコードとデータは公開されている」と直接宣言し [56]，Zhu らも GitHub 上のオープンソース成果物で検証を行い，replication package を提供している [58]．これらは，公開データが入手可能であるだけでなく，出版や再利用・引用を容易にする学術的な交換媒体として機能していることを示している．

**(3)「正解データ（Ground Truth)」の高い構築コスト**　TLR の実験評価には既知の真のリンクが必要である．しかし，リンクのアノテーション自体がドメイン知識と人手による確認に高度に依存しており，そのコストはプロジェクト規模に伴い急激に上昇する．Gotel らは，十分な自動化がない場合，トレーサビリティ関連活動は非常に高コストになると指摘している [13]．そのため，研究者は既にトレースマトリクスが存在するデータ，またはオープンソースリポジトリのルールから正解データを生成可能なデータ（例：コミットメッセージ，引用関係，リンクフィールド等を利用して Issue–Commit の正解を補助的に生成）を再利用する傾向がある．これは「公開データセット主導」という経路依存性をさらに強化する．

**(4) 研究の規模拡大（Scalability）への現実的需要**　既存のシステマティックマッピング研究は，現在の TLR 実証において規模と産業検証が依然として不足していると指摘しており，より多くの産業ケーススタディが必要であると述べると同時に，コンテキストやツールの詳細に関する報告にも改善の余地があるとしている [5]．この観察は，多くの研究が公開データに集中している理由が，単に公開データが重要だからではなく，それがクロスプロジェクト，大標本，かつ再現可能な実験デザインを支えやすいためであることを側面から説明している．

結論として，2011 年から 2025 年の間に「公開データセットのみ利用」が支配的であった核心的な理由は，研究者が産業界の問題を軽視しているからではない．産業成果物の機密性がデータ共有を制限し，学術コミュニティが再現性と比較可能性を強く求めていること，そして正解データの構築コストと規模拡大への需要が，研究をオープンソースおよび公開データへと強く推し進めた結果であると言える．

### 5.5　次元横断的な構造と研究の偏り（RQ4）

#### 5.5.1　なぜ主流リンク（Requirement–Code / Issue–Code）において特定の技術が結びつきやすいのか

**(1) Requirement–Code と IR の親和性**　まず，Requirement–Code は IR の「テキスト類似度 → ランキング」というパラダイムに本質的に適合している．多くの Requirement–Code シナリオにおいて，要求文書は自然言語であり，コード側も識別子やコメントなどのテキストシグナルを直接利用でき

る．そのため，IR の処理フロー（分かち書き，ストップワード除去，ステミング，類似度計算とランク付け）そのものが，解釈可能で参入障壁の低いベースライン生成パイプラインとして機能する [15]．同時に，このリンク種別における主な困難は，語彙の断絶や表現の不一致（「同じことを言っているが，用語が異なる」）に集中している．その結果，多くの改善提案は依然として IR の古典的な課題（Vocabulary Mismatch, Polysemy 等）に対する軽量な拡張を中心としている．例えば，IR の主流的地位を強調しつつも語彙の不一致による阻害を指摘する研究 [8] や，多義性による低精度を問題として提起し LSI で緩和を試みる研究 [73] などが挙げられる．これらの研究は，Requirement–Code において，研究者が IR という「再現可能なベースライン」を出発点とし，局所的な修正を加えるという経路を取りやすくし，結果として IR との結びつきを長期的に固定化している．

**(2) Issue–Code と ML/DL の結合**　一方で，Issue–Code が ML/DL と結びつきやすい理由は，表面的なテキスト類似度を超えた，意味的・構造的シグナルの融合が不可欠だからである．Issue 側は通常，変更や欠陥の記述であるが，Commit/Code 側にはコミットメッセージだけでなく，コードの変更内容やコンテキストが含まれる．Issue テキストとコミットログの類似度のみに依存すると，コードコンテキストという重要な変数が看過されやすい．ここに，「なぜ DL が必要か」という典型的な論証が登場する．すなわち，既存手法はテキスト類似度特徴に依存しコードコンテキストを無視しているが，コードコンテキストの意味論はリンク回復に重要な情報を提供するため，知識グラフを導入してコードコンテキストを表現し，その埋め込み（Embedding）を学習する必要があるという主張である [61]．同様に，BERT や CodeBERT のような事前学習済みモデルは，より強力な意味理解のために直接利用され，Issue テキスト，Commit テキスト，Commit コードを統合的にモデル化して分類やマッチングを行う [47]．また，自然言語（NL）とプログラミング言語（PL）の意味論的ギャップと，事前学習による転移の利点を強調し，深層モデルが大規模コードコーパスと追加の監督シグナルを利用できる点を指摘する研究もある [46]．これらの論証は，Issue–Code/Issue–Commit をより強力な表現学習が必要な領域へと押しやり，ML/DL との連携を密接にしている．

### 5.5.2　なぜデータセットの公開性がリンク種別によって分化するのか

**(1) 公開ベンチマークによる主流リンクの固定化**　IR ベースの TLR に関する体系的な整理において，評価に用いられる代表的なデータセット（CM-1, MODIS, EasyClinic 等）が CoEST によって継続的に提供されてきたことが指摘されている [5]．公開ベンチマークを持つリンク種別は，実験の再現，横断的な比較，および新手法の積み上げが容易であるため，研究はおのずと同一領域に集積する．その結果，公開ベンチマークのあるリンク種別では，「比較可能性の高さ→新手法の積み上げが容易→当該領域への更なる集中」という正のフィードバックが生じ，公開データの比率と研究上の主流化が同時に増幅される．さらに，多くの公開データセットが二部グラフ（Bipartite）構造を持っており，これが研究

を「2 種類の成果物間のリンク（Requirement–Code, Requirement–Design 等）」に集中させる基盤となり，特定の成果物間リンクにおいて公開データセットの比率が高くなる現象を説明している．

**(2) OSS 開発ログに起因する Issue–Code の公開データの優位性**　Issue–Code に関しては，OSS（オープンソースソフトウェア）において Issue トラッカーとコミット履歴（少なくともメタデータとして）が容易に公開されており，制度的にデータ収集が容易であるという特質がある．実際，Issue リンク研究の中には，「オープンソースからデータセットを収集した」と明記し，商用プロジェクトでの検証を将来の課題とするものがある [48]．このデータ取得の利便性は，本研究の結果（表 6）において Issue–Code が公開データに強く偏っている現象と一致している．

**(3) 上流成果物としての設計の機密性と非公開比率の上昇**　Requirement–Code や Issue–Code と比較して，Requirement–Design の研究では，「非公開/併用/自作」データセットが出現しやすい．これは設計成果物の生成環境とそこに含まれる情報の性質によって決定づけられる．設計モデル（BPMN，ステートマシン，制御モデル等）は，往々にしてビジネスプロセス，システム動作，セキュリティロジックを直接コード化しており，組織の核心的な技術資産である．また，これらは通常，企業内部のモデリングツールや管理ツールチェーン内で生成・保存される．そのため，関連研究の実証評価は，直接再利用可能な公開ベンチマークではなく，産業協力データや制限付きデータに依存する傾向が強い．

具体的な研究実践からも，この「産業内生データ」への依存構造が見て取れる．例えば BPMN に関するのシナリオでは，研究対象が産業パートナーの実際の要求とプロセスモデルに由来し，テキスト化データの導出さえも「社内ソリューション（in-house solution）」に依存しており，データセット（大量の要求/モデル要素を含む）自体が産業パートナーから提供されている例がある [66]．この種のデータは，オープンソースコードのように直接公開することは本質的に困難である（権利許諾，匿名化，フォーマット変換，保守が必要なため）．同時に，Requirement–Design リンクは人手によるアノテーションコストが高い（例えば，正解リンクの確立に多大な労力を要する）という問題にも直面しており，これが公開ベンチマーク構築の障壁をさらに高め，客観的に「公開データセットのみ」の比率を押し下げている．

### 5.6　研究および実務への提言

#### 5.6.1　研究面への提言

**(1) 研究の偏りを是正するための，ロングテールなリンク種別に向けた比較可能なベンチマークの構築**
本研究の RQ1 および RQ3 の結果から，研究が Requirement–Code と Issue–Code に高度に集中しており，その他のリンク種別（Requirement–Design，Design–Code，Document 関連等）は論文数が少なく，かつ公開データの入手可能性が著しく低いことが明らかになった．これは「主流リンク種別は公

開ベンチマーク上で反復的に改善され，比較可能な蓄積が形成される一方で，ロングテールなリンク種別は再現実験や横断的比較が困難であり，結果として研究規模と手法の収束速度がさらに抑制される」という構造的な帰結を招いている．したがって，今後の研究は，公開可能かつ再現可能なロングテール領域のベンチマーク構築（規模が小さくとも共有可能であること）に優先的に投資すべきである．同時に，データ共有の障壁とコンプライアンスリスクを低減するために，データの匿名化，抽象化，および配布プロトコルを明確に定義することが求められる．

**(2) 人間と計算機の協調コストの評価目標への統合**　主流の IR パラダイムは通常，候補リストを出力し，分析者に確認を委ねる．既存研究は「ユーザーフィードバックの伝播」「クエリ変換/書き換え」「マルチリトリーバー融合」などの方向で IR ランキングを継続的に最適化しているが，その共通の前提は「人間がループ内に存在し，システムの目的は単なる指標向上ではなく，人間のスクリーニング負担の軽減にある」という点である．そのため，今後の研究は，Top-k の審査に要する時間，確認コスト，誤検知（False Positive）がもたらす二次的コストをより体系的に報告し，リンク種別ごとにコスト対効果曲線（Cost-Benefit Curve）を提示すべきである．

**(3) データおよびシナリオ制約下における IR 手法に対する DL/LLM の「増分的利得の境界」の探求**　RQ2 の議論で述べた通り，学習ベースの手法はより強力な意味表現を動機としているが，その利得はデータ規模，ノイズ，およびプロジェクトの異質性に制約される．したがって，今後の研究においてより価値のある問いは，「どのような条件下（データ量，言語，成果物のスタイル，リンクの疎密度，クロスプロジェクト移行）において，学習ベースの手法が IR を上回るのか」，そして「いかにして IR やルールベースと組み合わせ，頑健な構成を形成するか」を明らかにすることである．

### 5.6.2　実務面への提言

**(1) 最も実用に近い導入形態：TLR を「全自動代替」ではなく「候補推薦＋人手確認」として位置づける**　多数の研究に共通するプロセスから見ると，現実的に実行可能な導入方法は，「システムが Top-k 候補リンクを生成し，それをレビューや変更プロセスに組み込み，エンジニアが確認して追跡資産として定着させる」という形態である．これは IR パラダイムのワークフローと高度に一致しており，IR が長期にわたり採用され続けている理由でもある．

**(2) Requirement–Code：解釈可能かつ導入障壁の低い IR パイプラインを優先し，軽量な拡張で典型的なボトルネックに対処する**　組織に比較的標準化された要求テキストとコードコメント/識別子が存在する場合，IR の類似度ランキングパイプラインは，最小の統合コストで実用的な候補を提供できる．実践においては，文献で成熟している軽量な拡張手法（多義性処理，クエリ変換，構造化テキスト

シグナルの利用，統計的スムージングなど）を優先的に採用し，用語の不一致やクエリ表現不足といった一般的な問題を緩和すべきである．

**(3) LLM への期待：判定器の代替だけではなく，「説明とインタラクション」への適用**　TLR における LLM の直接的な価値は，単なる「分類器の代替」として理解されるべきではない．より現実的な位置づけは，人手によるレビュープロセスにおける「説明，アライメント，およびインタラクション」への適用である．例えば，「要求–ゴールモデル」のような意味的関係が強いシナリオにおいて，LLM はリンクを出力するだけでなく，プロンプトの指示に従って各リンクに対する理由説明を生成できる．これにより，「なぜリンクされるのか/されないのか」を可読形式で分析者に提示し，人手による確認コストを低減させるとともに，監査やレビューをより運用可能なものにし，実工学上のコストを大幅に削減できる可能性がある．

### 5.7　本研究の限界と妥当性への脅威

本研究は系統的文献レビュー（SLR）であり，その結論は主に「検索—スクリーニング—コーディング—統計的解釈」という一連のプロセスに依存している．説明を明確にするため，本節では構成概念妥当性，内的妥当性，外的妥当性の 3 つの観点から，本研究の主要な限界を総括する．

#### 5.7.1　構成概念妥当性（Construct Validity）

**用語の境界によるバイアス**　TLR 関連研究には recovery, retrieval, generation, establishment など多様な表現が存在する．本研究は「TLR」を題目としているが，包含基準上は「追跡リンクの自動/半自動構築」を行う研究を一括して含めた．この処理は用語の差異による見落としを減らすことに寄与するが，同時にリスクも導入する．すなわち，異なる論文における「generation」や「recovery」のタスク設定や評価目的が完全に一致しない可能性があり，異質なタスクを含むことで，技術トレンドに対する解釈の精度に影響を与える可能性がある．

**分類軸の単純化による情報の損失**　本研究では技術パラダイムを IR / ML / DL / LLM / Other 等に，成果物を Requirement, Design, Code, Test, Issue, Document の 6 種に分類した．この抽象化は論文間の整合（アライメント）をとるのに有利だが，不可避的に詳細を圧縮する．例えば，同じ DL 研究でも全く異なるモデル構造や学習戦略を含みうるし，同一リンク種別の研究でも異なる具体的な対象に焦点を当てている可能性がある．これらの差異は集計において畳み込まれており，粒度の細かい差異が結論に与える影響を過小評価している可能性がある．

### 5.7.2 内的妥当性（Internal Validity）

**文献検索とスクリーニングのバイアス** 検索式は包含集合に直接影響する．もし検索語が「recovery」のみであれば，「generation」や「establishment」等の表現を用いる論文の再現率（Recall）が不足する可能性があり，逆もまた然りである．本研究では可能な限り包括的な検索クエリを設計したが，未知の用語を使用する研究を漏らしている可能性は排除できない．

**主観的判断による選定バイアス** タイトル/アブストラクトのスクリーニングおよび全文スクリーニング段階において，研究者による「追跡リンクタスクに該当するか」「自動/半自動リンク構築を満たすか」等の判断にはグレーゾーンが存在する．特にクロスドメイン応用（例：脆弱性レポートからファイルを特定する，モデル変更から Issue を追跡する等）の論文ではそれが顕著である．このような主観的判断は，異なるリンク種別の計数と分布に影響を与える可能性がある．

### 5.7.3 外的妥当性（External Validity）

**研究対象の OSS および公開データへの偏り** 第 4 章で示した通り，「公開データセットのみ利用」が支配的である．これは，本研究が「企業内部や機密性の高い成果物（特に設計成果物）」のシナリオに関する証拠が相対的に不足していることを意味する．したがって，「主流のリンク種別・主流技術」に関する本研究の結論は，「すべての産業シナリオにおける普遍的な事実」というよりは，公開データと再現可能な実験条件下における学術的な主流に近いものである．

**リンク種別のロングテールによる一般化の制限** サンプル数の少ないリンク種別においては，統計結果が個別の研究の影響を受けやすく，そこから安定した傾向を推論することは困難である．ロングテールなリンク種別に対しては，本研究は観測された現象と可能な原因を提示するにとどめ，強い断定を行うべきではない．

# 6　あとがき

本章では，本研究のまとめと今後の課題について述べる．

本研究は，ソフトウェアトレーサビリティリンク回復（Traceability Link Recovery: TLR）技術に関する研究動向を体系的に整理し，成果物ペア（リンク種別），技術パラダイム，およびデータセットの公開性の観点から，近年の研究構造と偏りを明らかにすることを目的として，系統的文献レビューを実施した．対象期間は 2011 年から 2025 年とし，最終的に 77 本の論文を選定・分析した．

分析の結果，まず RQ1（リンク種別）に関して，TLR 研究は主に Requirement–Code と Issue–Code に集中しており，その他のリンク種別は相対的に少数にとどまる傾向が確認された．次に RQ2（技術パラダイム）では，Requirement–Code では情報検索（IR）ベースの枠組みが長期にわたり中核を占める一方，Issue–Code では機械学習・深層学習（ML/DL）を中心とした学習ベース手法の比重が大きいことが示された．さらに RQ3（データセットの公開性）では，全期間を通じて公開データセットのみを用いた研究が支配的であり，特に主流リンク種別ほど公開データに基づく比較・再現が行われやすい構造が観察された．

加えて RQ4（次元横断）として，リンク種別・技術・データ入手性の間には共起傾向が存在し，主流リンク（Requirement–Code ／ Issue–Code）では「特定の技術パラダイム」への収束が生じやすい一方，ロングテールなリンク種別ではデータ取得の制約やドメイン依存性により，研究が分散しやすいことが示唆された．これらの結果は，TLR 研究が一定の成熟を見せる一方で，研究対象（リンク種別）と評価基盤（データセット）の偏りが，知見の一般化や比較可能性に影響し得ることを意味する．

今後の課題として，第一に，ロングテールなリンク種別に対する評価基盤の整備が重要である．設計成果物等の上流成果物は機密性が高く，公開データとして流通しにくいが，この制約が研究蓄積と手法比較の障壁になり得る．第二に，研究間比較可能性を高めるため，データセットの性質・前処理・タスク定義・評価手順の報告粒度を含む実験設計の透明化が求められる．第三に，実務導入の観点では，完全自動化よりも「候補推薦＋人手確認」を前提とした人間参加型ワークフローとしての位置付けが現実的であり，精度指標だけでなく運用コストや確認負荷を含む評価が必要である．第四に，LLM の活用については，判定の自動化そのものよりも，説明・対話・補助的推論として組み込み，機密情報・再現性・誤り（幻覚）に配慮したガバナンスと併用設計を行うことが重要となる．

最後に，本研究にはいくつかの限界がある．文献探索範囲と検索式に依存する選定バイアス，情報抽出および分類（リンク種別・技術パラダイム・データセットの公開性）の主観性，ならびに報告情報の不足に起因する不確実性は排除できない．これらの限界を踏まえつつ，本研究が TLR 研究の全体像を俯瞰し，将来の研究課題の設定と比較可能性向上に向けた議論の基盤となることを期待する．

## 謝辞

本研究の遂行にあたり，終始温かく，かつ熱心なご指導を賜りました 楠本 真二 教授に心より感謝申し上げます．先生の深い洞察と的確なご助言は，私の研究を導く道標となりました．また，研究生活における悩みに対しても親身に相談に乗っていただき，精神的な支えとなっていただいたことに深く感謝いたします．

本研究の全般にわたり，細部に至るまで丁寧かつ的確なご指導を頂きました 柗本 真佑 准教授に厚く御礼申し上げます．先生との議論を通じて得られた知見は，私の研究の質を高める上で極めて重要なものでした．

日頃の事務手続きをはじめ，研究室の円滑な運営にご尽力いただき，私を温かくサポートしてくださった事務の 橋本 美砂子 氏に心より御礼申し上げます．橋本様の細やかなお気遣いのおかげで，研究に専念できる環境が整っていました．

楠本研究室の同期である 岡本 琉生 氏，呉 梓靖 氏には，同じ研究グループのメンバーとして日々の議論や研究生活において多くの刺激を頂きました．また，忠谷 晃佑 氏，藪下 友 氏には，研究室の仲間として温かい励ましを頂きました．ここに感謝の意を表します．

中でも，チューターとして私を支えてくださった 玉置 文人 氏には，研究に関するアドバイスから日々の生活におけるサポートに至るまで，公私にわたり多大なるご協力を頂きました．献身的なサポートがなければ，本研究を完遂することは困難でした．ここに深く感謝いたします．

最後に，私の大学院生活を温かく見守り，精神的・経済的に支え続けてくれた家族に，深い感謝と敬愛の意を表します．

## 参考文献


[1] Winkler, S. and Pilgrim, von J.: A survey of traceability in requirements engineering and model-driven development, *Softw. Syst. Model.*, Vol. 9, No. 4, pp. 529–565 (2010).

[2] Cleland-Huang, J., Gotel, O. C. Z. and Zisman, A.: Software and Systems Traceability, in *Springer London* (2012).

[3] Diaz, D., Bavota, G., Marcus, A., Oliveto, R., Takahashi, S. and De Lucia, A.: Using code ownership to improve ir-based traceability link recovery, in *2013 21st international conference on program comprehension (icpc)*, pp. 123–132IEEE (2013).

[4] Ghabi, A. and Egyed, A.: Exploiting traceability uncertainty among artifacts and code, *J. Syst. Softw.*, Vol. 108, pp. 178–192 (2015).

[5] Borg, M., Runeson, P. and Ardö, A.: Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability, *Empir. Softw. Eng.*, Vol. 19, No. 6, pp. 1565–1616 (2014).

[6] Wang, B., Zou, Z., Liang, X., Jin, H. and Liang, P.: HGNNLink: recovering requirements-code traceability links with text and dependency-aware heterogeneous graph neural networks, *Automated Software Engineering*, Vol. 32, No. 2, p. 55 (2025).

[7] Hey, T., Chen, F., Weigelt, S. and Tichy, W. F.: Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations, in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 12–22 (2021).

[8] Gao, H., Kuang, H., Ma, X., Hu, H., Lü, J., Mäder, P. and Egyed, A.: Propagating frugal user feedback through closeness of code dependencies to improve IR-based traceability recovery, *Empirical Software Engineering*, Vol. 27, No. 2, p. 41 (2022).

[9] Sultanov, H. and Hayes, J. H.: Application of reinforcement learning to requirements engineering: requirements tracing, in *21st IEEE International Requirements Engineering Conference, RE 2013, Rio de Janeiro-RJ, Brazil, July 15-19, 2013*, pp. 52–61, IEEE Computer Society (2013).

[10] Guo, J., Cheng, J. and Cleland-Huang, J.: Semantically enhanced software traceability using deep learning techniques, in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pp. 3–14IEEE (2017).

[11] Lin, J., Liu, Y. and Cleland-Huang, J.: Information retrieval versus deep learning approaches for generating traceability links in bilingual projects, *Empirical Software Engineering*, Vol. 27,

No. 1, p. 5 (2022).

[12] Cleland-Huang, J., Gotel, O., Hayes, J. H., Mäder, P. and Zisman, A.: Software traceability: trends and future directions, in Herbsleb, J. D. and Dwyer, M. B. eds., *Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31 - June 7, 2014*, pp. 55–69, ACM (2014).

[13] Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P. and Antoniol, G.: The quest for Ubiquity: A roadmap for software and systems traceability research, in Heimdahl, M. P. E. and Sawyer, P. eds., *2012 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, USA, September 24-28, 2012*, pp. 71–80, IEEE Computer Society (2012).

[14] Mills, C.: Automating traceability link recovery through classification, in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pp. 1068–1070 (2017).

[15] Gao, H., Kuang, H., Sun, K., Ma, X., Egyed, A., Mäder, P., Rong, G., Shao, D. and Zhang, H.: Using consensual biterms from text structures of requirements and code to improve IR-based traceability recovery, in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1–1 (2022).

[16] Sommerville, I.: *Software Engineering*, Pearson, tenth edition (2016).

[17] Lyu, Y., Cho, H., Jung, P. and Lee, S.: A Systematic Literature Review of Issue-Based Requirement Traceability, *IEEE Access*, Vol. 11, pp. 13334–13348 (2023).

[18] White, R. and Krinke, J.: TCTracer: Establishing test-to-code traceability links using dynamic and static techniques, *Empirical Software Engineering*, Vol. 27, No. 3, p. 67 (2022).

[19] Nishikawa, K., Washizaki, H., Fukazawa, Y., Oshima, K. and Mibe, R.: Recovering transitive traceability links among software artifacts, in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 576–580IEEE (2015).

[20] Kitchenham, B.: *Kitchenham, B.: Guidelines for performing Systematic Literature Reviews in software engineering. EBSE Technical Report EBSE-2007-01* (2007).

[21] Okoli, C. and Schabram, K.: A Guide to Conducting a Systematic Literature Review of Information Systems Research, *SSRN Electronic Journal*, Vol. 10, (2010).

[22] Webster, J. and Watson, R. T.: Analyzing the Past to Prepare for the Future: Writing a Literature Review, *MIS Q.*, Vol. 26, No. 2 (2002).

[23] Guo, J. L. C., Steghöfer, J., Vogelsang, A. and Cleland-Huang, J.: Natural Language Processing for Requirements Traceability, *CoRR*, Vol. abs/2405.10845, (2024).

[24] Tange, W., Wang, C., Yao, P., Wu, R., Fu, X., Fan, G. and Zhang, C.: Dclink: Bridging data constraint changes and implementations in fintech systems, in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 914–925IEEE (2023).

[25] Lohar, S., Amornborvornwong, S., Zisman, A. and Cleland-Huang, J.: Improving trace accuracy through data-driven configuration and composition of tracing features, in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pp. 378–388 (2013).

[26] Mirakhorli, M., Shin, Y., Cleland-Huang, J. and Cinar, M.: A tactic-centric approach for automating traceability of quality concerns, in *2012 34th international conference on software engineering (ICSE)*, pp. 639–649IEEE (2012).

[27] Moran, K., Palacio, D. N., Bernal-Cárdenas, C., McCrystal, D., Poshyvanyk, D., Shenefiel, C. and Johnson, J.: Improving the effectiveness of traceability link recovery using hierarchical bayesian networks, in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 873–885 (2020).

[28] Florez, J. M., Perry, J., Wei, S. and Marcus, A.: Retrieving data constraint implementations using fine-grained code patterns, in *Proceedings of the 44th International Conference on Software Engineering*, pp. 1893–1905 (2022).

[29] Gao, H., Kuang, H., Assunção, W. K., Mayr-Dorn, C., Rong, G., Zhang, H., Ma, X. and Egyed, A.: Triad: Automated traceability recovery based on biterm-enhanced deduction of transitive links among artifacts, in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13 (2024).

[30] Fuchß, D., Hey, T., Keim, J., Liu, H., Ewald, N., Thirolf, T. and Koziolek, A.: LiSSA: toward generic traceability link recovery through retrieval-augmented generation, in *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering. ICSE*, Vol. 25 (2025).

[31] Mills, C., Escobar-Avila, J. and Haiduc, S.: Automatic Traceability Maintenance via Machine Learning Classification, in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 369–380 (2018).

[32] Mills, C., Escobar-Avila, J., Bhattacharya, A., Kondyukov, G., Chakraborty, S. and Haiduc, S.: Tracing with Less Data: Active Learning for Classification-Based Traceability Link Recovery, in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 103–113 (2019).

[33] Rodriguez, A. D., Cleland-Huang, J. and Falessi, D.: Leveraging Intermediate Artifacts to Improve Automated Trace Link Retrieval, in *IEEE International Conference on Software*

*Maintenance and Evolution, ICSME 2021, Luxembourg, September 27 - October 1, 2021*, pp. 81–92, IEEE (2021).

[34] Omoronyia, I., Sindre, G. and Stålhane, T.: Exploring a Bayesian and linear approach to requirements traceability, *Inf. Softw. Technol.*, Vol. 53, No. 8, pp. 851–871 (2011).

[35] Strasunskas, D. and Hakkarainen, S.: Domain model-driven software engineering: A method for discovery of dependency links, *Inf. Softw. Technol.*, Vol. 54, No. 11, pp. 1239–1249 (2012).

[36] Lucia, A. D., Penta, M. D., Oliveto, R., Panichella, A. and Panichella, S.: Applying a smoothing filter to improve IR-based traceability recovery processes: An empirical investigation, *Inf. Softw. Technol.*, Vol. 55, No. 4, pp. 741–754 (2013).

[37] Bavota, G., Lucia, A. D., Oliveto, R. and Tortora, G.: Enhancing software artefact traceability recovery processes with link count information, *Inf. Softw. Technol.*, Vol. 56, No. 2, pp. 163–182 (2014).

[38] Ali, N., Cai, H., Hamou-Lhadj, A. and Hassine, J.: Exploiting Parts-of-Speech for effective automated requirements traceability, *Inf. Softw. Technol.*, Vol. 106, pp. 126–141 (2019).

[39] Blasco, D., Cetina, C. and Pastor, O.: A fine-grained requirement traceability evolutionary algorithm: Kromaia, a commercial video game case study, *Inf. Softw. Technol.*, Vol. 119, (2020).

[40] Mondal, A. K., Hossain, M., Roy, C. K., Roy, B. and Schneider, K. A.: FSECAM: A contextual thematic approach for linking feature to multi-level software architectural components, *J. Syst. Softw.*, Vol. 219, p. 112245 (2025).

[41] Li, J., Liu, S. and Jin, Z.: Automated formal-specification-to-code trace links recovery using multi-dimensional similarity measures, *J. Syst. Softw.*, Vol. 226, p. 112439 (2025).

[42] Niu, N. and Mahmoud, A.: Enhancing candidate link generation for requirements tracing: The cluster hypothesis revisited, in Heimdahl, M. P. E. and Sawyer, P. eds., *2012 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, USA, September 24-28, 2012*, pp. 81–90, IEEE Computer Society (2012).

[43] Mahmoud, A.: An information theoretic approach for extracting and tracing non-functional requirements, in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 36–45 (2015).

[44] Kuang, H., Nie, J., Hu, H., Rempel, P., Lu, J., Egyed, A. and Mäder, P.: Analyzing closeness of code dependencies for improving IR-based Traceability Recovery, in Pinzger, M., Bavota, G. and Marcus, A. eds., *IEEE 24th International Conference on Software Analysis, Evolution and*

*Reengineering, SANER 2017, Klagenfurt, Austria, February 20-24, 2017*, pp. 68–78, IEEE Computer Society (2017).

[45] Ali, N., Guéhéneuc, Y. and Antoniol, G.: Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links, *IEEE Trans. Software Eng.*, Vol. 39, No. 5, pp. 725–741 (2013).

[46] Zhang, C., Wang, Y., Wei, Z., Xu, Y., Wang, J., Li, H. and Ji, R.: EALink: An efficient and accurate pre-trained framework for issue-commit link recovery, in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 217–229IEEE (2023).

[47] Lan, J., Gong, L., Zhang, J. and Zhang, H.: BTLink: automatic link recovery between issues and commits based on pre-trained BERT model, *Empirical Software Engineering*, Vol. 28, No. 4, p. 103 (2023).

[48] Wu, R., Zhang, H., Kim, S. and Cheung, S.-C.: Relink: recovering links between bugs and changes, in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 15–25 (2011).

[49] Nguyen, A. T., Nguyen, T. T., Nguyen, H. A. and Nguyen, T. N.: Multi-layered approach for recovering links between bug reports and fixes, in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, pp. 1–11 (2012).

[50] Dong, L., Zhang, H., Liu, W., Weng, Z. and Kuang, H.: Semi-supervised pre-processing for learning-based traceability framework on real-world software projects, in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 570–582 (2022).

[51] Deng, Y., Wang, B., Zou, Z. and Ye, L.: PromptLink: Multi-template prompt learning with adversarial training for issue-commit link recovery, in *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 461–467 (2024).

[52] Le, T.-D. B., Linares-Vásquez, M., Lo, D. and Poshyvanyk, D.: Rclinker: Automated linking of issue reports and commits leveraging rich contextual information, in *2015 IEEE 23rd international conference on program comprehension*, pp. 36–47IEEE (2015).

[53] Rath, M., Rendall, J., Guo, J. L., Cleland-Huang, J. and Mäder, P.: Traceability in the wild: automatically augmenting incomplete trace links, in *Proceedings of the 40th International Conference on Software Engineering*, pp. 834–845 (2018).

[54] Lin, J., Liu, Y., Zeng, Q., Jiang, M. and Cleland-Huang, J.: Traceability transformed: Gener-

ating more accurate links with pre-trained bert models, in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 324–335IEEE (2021).

[55] Sun, J., Chen, J., Xing, Z., Lu, Q., Xu, X. and Zhu, L.: Where is it? tracing the vulnerability-relevant files from vulnerability reports, in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13 (2024).

[56] Mazrae, P. R., Izadi, M. and Heydarnoori, A.: Automated Recovery of Issue-Commit Links Leveraging Both Textual and Non-textual Data, in *IEEE International Conference on Software Maintenance and Evolution, ICSME 2021, Luxembourg, September 27 - October 1, 2021*, pp. 263–273, IEEE (2021).

[57] Zhu, J., Xiao, G., Zheng, Z. and Sui, Y.: Enhancing Traceability Link Recovery with Unlabeled Data, in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 446–457 (2022).

[58] Zhu, J., Xiao, G., Zheng, Z. and Sui, Y.: Deep semi-supervised learning for recovering traceability links between issues and commits, *J. Syst. Softw.*, Vol. 216, p. 112109 (2024).

[59] Wang, B., Deng, Y., Luo, R., Liang, P. and Bi, T.: MPLinker: Multi-template Prompt-tuning with adversarial training for Issue-commit Link recovery, *J. Syst. Softw.*, Vol. 223, p. 112351 (2025).

[60] Liu, Y., Lin, J. and Cleland-Huang, J.: Traceability Support for Multi-Lingual Software Projects, in Kim, S., Gousios, G., Nadi, S. and Hejderup, J. eds., *MSR '20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29-30 June, 2020*, pp. 443–454, ACM (2020).

[61] Xie, R., Chen, L., Ye, W., Li, Z., Hu, T., Du, D. and Zhang, S.: DeepLink: A Code Knowledge Graph Based Deep Learning Approach for Issue-Commit Link Recovery, in Wang, X., Lo, D. and Shihab, E. eds., *26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February 24-27, 2019*, pp. 434–444, IEEE (2019).

[62] Pârtachi, P., White, D. R. and Barr, E. T.: Aide-mémoire: Improving a Project's Collective Memory via Pull Request-Issue Links, *ACM Trans. Softw. Eng. Methodol.*, Vol. 32, No. 2, pp. 32:1–32:36 (2023).

[63] Bai, S., Liu, H., Dai, E. and Liu, L.: Improving Issue-PR Link Prediction via Knowledge-Aware Heterogeneous Graph Learning, *IEEE Trans. Software Eng.*, Vol. 50, No. 7, pp. 1901–1920 (2024).

[64] Hassine, J.: An llm-based approach to recover traceability links between security requirements and goal models, in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pp. 643–651 (2024).

[65] Alenazi, M., Niu, N. and Savolainen, J.: A novel approach to tracing safety requirements and state-based design models, in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 848–860 (2020).

[66] Lapeña, R., Pérez, F., Pastor, O. and Cetina, C.: Leveraging execution traces to enhance traceability links recovery in BPMN models, *Inf. Softw. Technol.*, Vol. 146, p. 106873 (2022).

[67] Marcén, A. C., Lapeña, R., Pastor, O. and Cetina, C.: Traceability Link Recovery between Requirements and Models using an Evolutionary Algorithm Guided by a Learning to Rank Algorithm: Train control and management case, *J. Syst. Softw.*, Vol. 163, p. 110519 (2020).

[68] Saini, R., Mussbacher, G., Guo, J. L. C. and Kienzle, J.: Automated Traceability for Domain Modelling Decisions Empowered by Artificial Intelligence, in *29th IEEE International Requirements Engineering Conference, RE 2021, Notre Dame, IN, USA, September 20-24, 2021*, pp. 173–184, IEEE (2021).

[69] Dietrich, T., Cleland-Huang, J. and Shin, Y.: Learning effective query transformations for enhanced requirements trace retrieval, in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 586–591IEEE (2013).

[70] Sultanov, H.: Requirements tracing: discovering related documents through artificial pheromones and term proximity, in *Proceedings of the 33rd International Conference on Software Engineering*, pp. 1173–1175 (2011).

[71] Rajpathak, D., Peranandam, P. M. and Ramesh, S.: Automatic development of requirement linking matrix based on semantic similarity for robust software development, *J. Syst. Softw.*, Vol. 186, p. 111211 (2022).

[72] Gervasi, V. and Zowghi, D.: Supporting traceability through affinity mining, in Gorschek, T. and Lutz, R. R. eds., *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, pp. 143–152, IEEE Computer Society (2014).

[73] Wang, W., Niu, N., Liu, H. and Niu, Z.: Enhancing Automated Requirements Traceability by Resolving Polysemy, in Ruhe, G., Maalej, W. and Amyot, D. eds., *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, pp. 40–51, IEEE Computer Society (2018).

[74] Schlutter, A. and Vogelsang, A.: Trace Link Recovery using Semantic Relation Graphs and

Spreading Activation, in Breaux, T. D., Zisman, A., Fricker, S. and Glinz, M. eds., *28th IEEE International Requirements Engineering Conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020*, pp. 20–31, IEEE (2020).

[75] Keim, J., Corallo, S., Fuchß, D., Hey, T., Telge, T. and Koziolek, A.: Recovering Trace Links Between Software Documentation And Code, in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13 (2024).

[76] Liu, Y., Lin, J., Zeng, Q., Jiang, M. and Cleland-Huang, J.: Towards Semantically Guided Traceability, in Breaux, T. D., Zisman, A., Fricker, S. and Glinz, M. eds., *28th IEEE International Requirements Engineering Conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020*, pp. 328–333, IEEE (2020).

[77] Mirakhorli, M. and Cleland-Huang, J.: Detecting, Tracing, and Monitoring Architectural Tactics in Code, *IEEE Trans. Software Eng.*, Vol. 42, No. 3, pp. 206–221 (2016).

[78] White, R., Krinke, J. and Tan, R.: Establishing multilevel test-to-code traceability links, in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 861–872 (2020).

[79] Qusef, A., Bavota, G., Oliveto, R., Lucia, A. D. and Binkley, D. W.: Recovering test-to-code traceability using slicing and textual analysis, *J. Syst. Softw.*, Vol. 88, pp. 147–168 (2014).

[80] Sun, W., Guo, Z., Yan, M., Liu, Z., Lei, Y. and Zhang, H.: Method-Level Test-to-Code Traceability Link Construction by Semantic Correlation Learning, *IEEE Trans. Software Eng.*, Vol. 50, No. 10, pp. 2656–2676 (2024).

[81] Trubiani, C., Ghabi, A. and Egyed, A.: Exploiting traceability uncertainty between software architectural models and extra-functional results, *J. Syst. Softw.*, Vol. 125, pp. 15–34 (2017).

[82] Xiaofan Chen, X. and Grundy, J.: Improving automated documentation to code traceability by combining retrieval techniques (2011).

[83] Dagenais, B. and Robillard, M. P.: Recovering traceability links between an API and its learning resources, in *2012 34th international conference on software engineering (icse)*, pp. 47–57IEEE (2012).

[84] Gu, J., Wen, J., Wang, Z., Zhao, P., Luo, C., Kang, Y., Zhou, Y., Yang, L., Sun, J., Xu, Z., et al.: Efficient customer incident triage via linking with system incidents, in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1296–1307 (2020).

[85] Sun, J., Xing, Z., Xu, X., Zhu, L. and Lu, Q.: Heterogeneous Vulnerability Report Traceabil-

ity Recovery by Vulnerability Aspect Matching, in *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 175–186 (2022).

[86] Kim, J. and Hong, S.: Inferring Fine-grained Traceability Links between Javadoc Comment and JUnit Test Code, in *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 424–428 (2022).

[87] Oosten, van W., Rasiman, R., Dalpiaz, F. and Hurkmans, T.: On the effectiveness of automated tracing from model changes to project issues, *Inf. Softw. Technol.*, Vol. 160, p. 107226 (2023).

[88] Gadelha, G., Ramalho, F. and Massoni, T.: Traceability recovery between bug reports and test cases-a Mozilla Firefox case study, *Automated Software Engineering*, Vol. 28, No. 2, p. 8 (2021).

[89] Pauzi, Z. and Capiluppi, A.: Applications of natural language processing in software traceability: A systematic mapping study, *Journal of Systems and Software*, Vol. 198, p. 111616 (2023).