様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　６　月　１９　日現在

機関番号：　１２６１２
研究種目：　挑戦的萌芽研究
研究期間：　2014～2016
課題番号：　２６５４０１６９

研究課題名（和文）ソフトウェア理解ドメインと一体的に共発展するプログラム大辞典

研究課題名（英文）Coevolution of code feature dictionary and software comprehension domain

研究代表者
前川　守（Maekawa, Mamoru）

電気通信大学・その他部局等・名誉教授

研究者番号：１０１２６１６２

交付決定額（研究期間全体）：（直接経費）　　2,800,000円

研究成果の概要（和文）：技術者によるソースコード理解を容易にするためのシステム支援として，ソースコードに対するアノテーションと検索を一体的に扱うことによって，複数の利用者が自らの知識をソースコードと明確に対応付けてシステムに蓄積していき，必要に応じて対話的な一般化によってソースコード内での類似個所をシステムが提示する等が可能な枠組みを提案した．
その実現手法として，抽象構文木のノードタプルをソースコード上の意味付与対象とする事，対象の一般化をプリミティブ属性関数の再帰的組合せで構成する検索手法空間での判別で実現する事，発見的・適応的に探索のバイアスを変更する事等を提案し，プロトタイプの実装と評価を行なった．

研究成果の概要（英文）：Understanding unfamiliar source code is inherently difficult for a software engineer, despite its importance. It is, however, still hard for a system to help in this activity, for lack of ways of both representing semantic chunks and of preparing a rich dictionary of chunks.
This research proposed an integrated framework for annotating and searching source code. In the framework, each engineer gathers (annotates) semantic chunks that have the same meaning and interactively generalizes them to get a search pattern. As a result, a dictionary of semantic chunks together with their search patterns is incrementally created through engineer collaboration. Two representations are used: a tuple of nodes of an abstract syntax tree (AST) for a semantic chunk and a classifier on generative attribute vectors for search patterns.

研究分野：　ソフトウェア工学

キーワード：　抽象構文木　検索　アノテーション

１．研究開始当初の背景

(1) ソフトウェアの大規模蓄積に内在するソフトウェア知は未利用.

多くのソフトウェアが開発され続け，その蓄積も増え続けている.

今後作られるソフトウェアも，総体としては既存のものと似た傾向があると考えられ，蓄積が大規模になればなるほど，類似のものが既に存在する可能性が高まる．すなわち，蓄積されているソフトウェア知の潜在的な利用価値も高まる.

全世界を対象とした大規模なソフトウェア蓄積に埋もれているソフトウェア知が有効利用できれば，無駄な再開発がなくなり，より本質的で革新的な開発に社会資源を集中できる．蓄積から得た既存ソフトウェアの再利用で開発が容易になるだけでなく，従来から改良されてきたものの方が誤りも少なく高効率でロバストである.

(2) ソフトウェア検索の困難さとその理由.

ソフトウェア知を有効利用するためには，蓄積から必要な知を的確に検索して理解する事が容易でなければならない.

しかし，そのようなソフトウェア検索技術は確立されていない．主因は次の点である.

① 多くの手法が未探究.

未だ多くの手法が未探究である．汎用で使える適用範囲の広い手法だけでなく，限定された対象や文脈でのみに有効な手法なども網羅的に探究すべきである.

② 従来の各種手法と実用場面での検索意図が不整合.

ソフトウェア検索が実用というよりまだ研究中心ということもあり，従来手法は原理追求に重点があった.

一方，実際に使用する技術者の立場からは，どの単一の原理も自らの用途に万能ではない以上，各種手法を適切に組み合わせたり使い分けたりして自らの検索を実現したいが，それができなかった.

③ 検索意図や背景知識の表現手段が不足.

同様の理由で，従来手法は「ある特定の検索意図はこの原理で満たされる」という，言わば原理ありきの形であり，技術者が検索意図や背景知識を的確に表現する手段は未探究だった.

２．研究の目的

(1) ソフトウェア検索手法の網羅的表現による拡張.

既存手法のみならず，凡そソフトウェア検索と言える全てのものが網羅されている空間があったとしよう．その中から開発者が検索意図に最適な手法を容易に探して利用できれば，検索手法空間を最大限に活かす事になる.

そこで，ソフトウェアの一部や全体の特徴として考えられるものを細粒度まで含めて網羅し，それらの組合せで従来のソフトウェア理論・ソフトウェア工学・アルゴリズム論等の知見を一様に表現あるいは近似するとともに，さらに大きく，または，形を変えて延長する.

その結果としてプログラム検索・ソフトウェア開発を全世界的・全分野的・全面的に支援する.

ソフトウェアの各種特徴を要素として構成する事によりソフトウェア検索手法を網羅的に扱う点が，このアプローチの特色である.

(2) 研究期間内の目標.

① 検索手法空間.

既存の各種検索手法を含むできるだけ多くの手法を，表現あるいは近似できるような検索手法空間の構成法を提案する.

検索手法空間には，原理をストレートに反映した汎用の手法のみならず，特定の検索状況のみに適用可能な手法や，原理的には正しさが示せないような近似手法(ヒューリスティクス)も含められるようにする.

② 検索意図および背景知識の表現法.

しかし，検索手法空間は極めて広く，実用上無意味な多くの手法も含まれる.

そこで，利用者である技術者による検索手法探索を容易にするために，検索手法空間に，開発者の検索意図や背景知識を反映して無意味なものを効果的に除外する構造や探索時の優先順位を表現し得るようにする.

３．研究の方法

(1) 多種のソフトウェア特徴の組合せによる検索手法空間の構成法.

細粒度のものも含む多種類のソフトウェア特徴を列挙し，それらを要素として組み合せる事によって，各種の検索手法を実現または近似的に実現できる検索手法空間を構成する.

この空間の個々の要素は，それぞれが 1 つの検索手法に対応し，ソフトウェア特徴の組み合せを変えた多くの検索手法が網羅的・一様に表現される．検索手法空間には，既存手法のみならず潜在的に多くの未知で有益な手法が含まれると期待される.

枠組みは対象言語や使用する特徴の種類に依存しないため，広く利用されている Java 言語を対象とした．また，プロジェクトのコード全体が利用できなかったり，コードに誤りがあったりするなどの理由で実行できない場合も利用できるよう，ソースコードの静的解析で得られる特徴を用いた.

(2) プロトタイプシステムの作成と実験.

プロトタイプシステムの実装も Java 言語で行なった．オープンソースの統合開発環境のプラグインとして実現し，統合開発環境自体が持つ構文解析等のクラスライブラリを利用した.

実験は，技術者が自らの検索意図に対して期待する結果の例を複数与え，どの検索手法で

探せる可能性があるかの候補をシステムに帰納的に推測させる事を基本とした．この実験は，次項の漸進的な改良のための知見を得る事が大きな目的であり，繰り返し行なった．なお，テストデータとしてオープンソースのソフトウェアを用いた．

(3) 検索意図および背景知識を容易に反映し得る探索方法の提案と漸進的改良．
結果例の類似性により検索手法の類似性を推測し，そこから摂動するなどの探索ヒューリスティクスや，技術者が理解している事をできるだけシステムに伝え易くするための対話的なフィードバックなどにより，検索手法空間内の探索におけるヒューリスティックなバイアスをシステムが構築する手法を提案した．

４．研究成果
(1) 想定する利用形態の提案．
① アノテーションと検索の一体的取扱い．
技術者はソースコードに対する作業過程で得る断片的な知識を，ソースコードの部分に対するアノテーションとして蓄積していく．アノテーションには，自然言語の短文で表したタグを付与できるようにした．
アノテーションを付与した部分は何らかの意味のまとまりを持っていると考えられ，構文として似た部分は，同じアノテーションを付与できる可能性が高い．
そこで，アノテーションを付与された部分と類似した他の部分を対話的に検索できることとした．
② 技術者による対話的なフィードバック．
検索意図や背景知識を反映させるため，技術者がシステムを対話的に誘導するインタラクションとして，ソースコード内の関連部分を示唆する方法と，優先的に考慮すべき特徴を示唆する方法を提案した．
③ 複数利用者による協調．
この枠組みは，ソースコードの表層的な意味をシステムに教えていく過程と見なせる．多くの利用者が協調するほど，定義されるアノテーションの網羅範囲と精度は高まっていくと期待される．
そこで，蓄積された結果例や検索手法を，必要であれば選択的に利用することで，たとえ技術者間でアノテーション付与に矛盾があったとしても複数技術者による協調が可能な方法を提案した．

(2) 手法の提案．
① ソースコードの部分の表現方法．
文法に適合したソースコードの構文解析結果は，抽象構文木(Abstract Syntax Tree)で表現できる．そこで，アノテーションの付与対象および検索結果の表現として，抽象構文木のノードの組を使う事を提案した．
技術者を対象とした実験により，この表現がアノテーションや検索結果として自然に利用できる形である事が示唆された．
② ノードの組の属性値空間へのマッピング．
個々の検索手法を，抽象構文木のノードの組に対する直接的な形ではなく，属性値ベクトルへマッピングした後の空間で表現する事を提案した．
これにより，抽象構文木のノードの組に対して，例えば，識別子名の特徴，構文上の特徴，パッケージやクラス階層の関係，データフローや制御フローの特徴など，どのような属性値を設定するかによって，表現できる検索手法の種類を変えられる．また，抽象構文木の木構造を直接扱う場合と比べて，属性値ベクトルという単純で一様な構造になるため，探索が制御し易くなった．
③ プリミティブ属性関数の組み合せによる属性関数ベクトル生成．
検索手法空間は，ヒューリスティックな手法等も含んで網羅的でなければならない．大きな空間を一様・体系的に構成するために，ソフトウェアの各種特徴を要素とし，それを基に手法を構成する．
そこで，属性値ベクトルを，プリミティブ属性関数の再帰的な組合せによって生成することを提案した．概念的には再帰的な総当たりの全体が検索手法空間となるが，実際には後述する検索意図および背景知識の表現によって，実用上意味のある範囲に限定する．
プリミティブ属性関数は，従来の各種手法のアルゴリズムの過程で使われる特徴群を含むのは勿論のこと，抽象構文木のノードについての属性を求めるものや，さらにその結果に対して適用できるものなどを含む．例えば，識別子同士の類似性を判定したり，データフローを遡って抽象構文木の他のノードを得たりするようなものがあり，これらの組合せによって検索手法を一様に表現できる．これが検索手法空間を構成する．
プロトタイプシステムでは，プリミティブ属性関数は Java のメソッドとして実装し，抽象構文木のノードも含め，全ての属性値を Java のオブジェクトとして表現した．
④ 正例・負例の判別による検索手法の推測．
技術者から与えられたアノテーションを正例として，また，検索結果が意図したものと異なる旨のフィードバックを技術者から与えられた部分を負例として，検索手法空間上で判別を行ない，得られた判別規則を検索手法の近似とした．
説明のため単純な例を挙げる．「同一の配列の要素同士の交換がある」「それを含む繰り返しがある」「その外側に，ある変数 h を減ずる繰り返しがある」「その繰り返しの前に，配列のサイズに依存して h に代入する式がある」などの断片的な構文上の特徴が属性値ベクトル上で判別され，これが Shell ソートである可能性が高いと推測される．
判別手法は，プロトタイプシステムでは主に判別木を用いたが，識別子の類似度のような連続的な値も反映し得るようにするための

手法も模索した.
⑤ 検索意図および背景知識の表現.
プリミティブ属性関数を組み合せて属性値ベクトルを生成する際にどれを優先するかや，既に生成された属性値ベクトルの要素のうちどれを優先して判別するかを制御することによって，検索意図をより的確に反映した推測を行なったり，背景知識を反映して無駄な探索を抑制したりする方法を提案した.
どれを優先するかは，技術者からの対話的なフィードバックから得るほか，既存の成功した判別例から，プリミティブ属性関数間やそれらの結果の値の間に成立すると推測される関係を徐々に獲得する方法を提案した.
例えば，ほとんど常に同じ結果となる属性値や，複数の属性値の組み合わせでほとんど常に推測できるものは，アソシエーションルールマイニング等の手法で検出して除外した.
逆に，判別への寄与の大きい属性値やプリミティブ属性関数は優先した.

(3) 実験による評価.
小規模ではあるが技術者による実験を行ない，提案した手法および枠組みが実用可能性のあるものである示唆を得た.
一方，技術者が実用上許容できる程度にインタラクションを抑えることや，システムによる探索の計算量を実用的な範囲に抑えることなどに関連して，さらに改善すべき点についての知見も得た.

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 0 件）

〔学会発表〕（計 2 件）

① Ken Nakayama, Shun’ichi Tano, Tomonori Hashiyama, Eko Sakai, Incremental annotate-generalize-search framework for interactive source code comprehension, Proc. of IEEE COMPSAC 2017, 7 Jul 2017, Turin (Italy), 査読有 (to appear)

② Ken Nakayama, Eko Sakai, Michihiro Kobayakawa, Interactive characterization of a code pattern, Proc. of the 13th International Conference on Intelligent Software Methodologies, Tool, and Technique (SoMeT 2014), pp. 1016-1025, 24 Sep 2014, Langkawi (Malaysia), 査読有

６．研究組織

(1)研究代表者
前川 守（MAEKAWA, Mamoru）
電気通信大学・その他部局等・名誉教授
研究者番号： 10126162

(2)研究分担者
大須賀 昭彦（OHSUGA, Akihiko）
電気通信大学・大学院情報理工学研究科・教授
研究者番号： 90393842

(3)研究分担者
川村 隆浩（KAWAMURA, Takahiro）
電気通信大学・大学院情報理工学研究科・客員准教授
研究者番号： 10426653

(4)研究分担者
中山 健（NAKAYAMA, Ken）
津田塾大学・数学・計算機科学研究所・専任研究員
研究者番号： 40296348