科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　　７　日現在

機関番号：　１４４０１
研究種目：　基盤研究(A)
研究期間：　2009～2013
課題番号：　２１２４０００２

研究課題名（和文）巨大ソフトウェア工学データを対象とした計算ソフトウェア工学の確立

研究課題名（英文）Establishing Calculation Software Engineering for Big Data of Software Engineering

研究代表者
井上　克郎（Katsuro, Inoue）

大阪大学・情報科学研究科・教授

研究者番号：２０１６８４３８

交付決定額（研究期間全体）：（直接経費）　　35,500,000円、（間接経費）　　10,650,000円

研究成果の概要（和文）：種々のソフトウェア開発プロジェクトに関するソースコードなどのデータを多数集め、計算機資源を用いてソフトウェア開発や保守に関する普遍的な知識を得る計算ソフトウェア工学の確率をめざし種々の研究を行った。具体的には、ソフトウェアの自動推薦、キャップを含むクローン検出法、ファイルクローン検出法などの成果等を得た。

研究成果の概要（英文）：We established calculation software engineering, applying software engineering techniques to big data that comes from lots of software developing projects. Contributions include auto-recommendation of components, detecting gapped clones, and detecting file clones.

研究分野：　総合領域

科研費の分科・細目：　情報学・ソフトウェア

キーワード：　コードクローン　ソフトウェア構成関連分析　実証データ分析

１．研究開始当初の背景

ソフトウェア工学とは、ソフトウェアの生産性や品質の向上を目指す諸技術で、いろいろな方法やツールが提案され、実際に使われてきている。そして、その技術の多くは、主に小規模な開発作業を対象としており、ソースコードやドキュメント、開発プロセスデータ等の比較的小規模な集合に対して、効率よく検索、分析、変更等の作業を行うことを支援している。

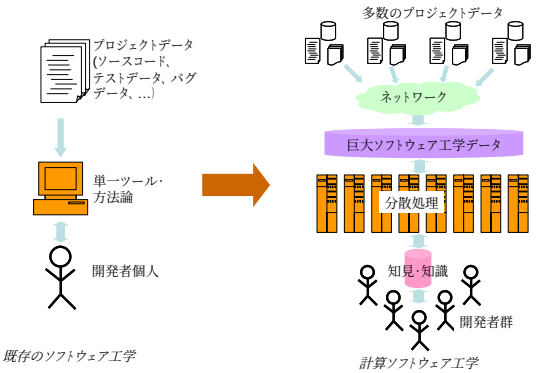
近年、ローカルネットワークからインターネットまで、種々のネットワークの性能の向上は著しい。これらのネットワークを通じて、過去に開発された種々のソフトウェアシステムのソースコードや設計書、テストデータや実証データ等のいわゆるソフトウェア工学データを収集することは容易で、個人の作業データはもとより、組織全体や、組織の枠を超えたデータを集めることもできるようになってきた。

このような環境の変化に適応して、種々のソフトウェア開発プロジェクトに関するソフトウェア工学データを多数集め、それを深く解析することによって、ソフトウェアの生産や保守に関する普遍的な知識が得られることが期待される。また、近年の計算機システムのハードウェアは高性能で安価になり、多コア CPU の利用や、複数の PC でクラスタやグリッドシステムを構築することが容易になってきた。これらの計算機ハードウェアで並列処理や分散処理を行い、巨大なソフトウェア工学データを効率的に処理し、ソフトウェアの品質や生産性の向上につなげていくことが可能となっている。

既存のソフトウェア工学の諸問題においては、並列処理や分散処理を用いている例はほとんどない。これは、一般にソフトウェア工学の問題は、計算対象の依存関係が複雑で、効率よく計算を行えるよう、複数に分割することが容易ではないと思われていたからである。しかし、我々は、大規模なソフトウェア工学データを扱う研究を通じて、問題分割が可能な例を発見してきた。例えば、オープンソースシステムの変遷を調べるために、大規模コードクローン分析を行ってきたが、この問題は、EP 問題（Embarrassingly Parallel problems、並列化が自明な問題）であることがわかり、このための分散処理システムを開発し、巨大なデータの分析を行った。

２．研究の目的

このような背景に基づき、より高い品質のソフトウェアを高い生産性で開発できるようにするために、大規模なソフトウェア工学データに対して深い分析を行うソフトウェア工学技術「計算ソフトウェア工学」を確立する。



そのために、いくつかのソフトウェア工学分野の問題にテーマに絞り、手法の開発、ツールの作成、大規模ソフトウェア工学データの分析等を行い、計算ソフトウェア工学の実証例の知見を蓄積するとともに、計算ソフトウェア工学のためのデータモデルや計算モデルの開発を行う。具体的には、次の(1)～(3)のサブテーマと（4）の統合テーマの研究を行う。

(1) 分散コードクローン分析
分散処理システムやマルチコア CPU システムで効率よく稼動するコードクローン分析手法を開発し、オープンソースシステムとして開発し、公開する。開発したシステムを用いて、ソフトウェアシステム間の相関やソフトウェア著作物の再利用状況の分析を行う。

(2) ソフトウェアシステム間の大規模関連分析
多数のソフトウェアシステム間の関連を知るために、主にソースコードの間の関連をモデル化して分析し可視化する手法を開発するとともに、実用的なソフトウェアシステム閲覧システムの開発を行う。

(3) 大規模実証データ収集、分析
ソフトウェア開発に関わるプロセスや生産物のデータ（実証データ）をプロジェクトにまたがって多量に集めて蓄積し、それを分散処理システムで効率よく分析し、開発者にフィードバックするプロトタイプシステムを開発する。

(4) 計算ソフトウェア工学モデルの構築
上記(1)～(3)での知見を元にして、計算ソフトウェア工学で頻繁に用いられるデータ構造や計算方式を一般化したモデルを構築するとともに、それを用いたソフトウェア工学の諸問題の定義方法や、そのモデルを効率よく実行するための分散計算システムのアーキテクチャの設計とプロトタイプシステムの作成を行う。

３．研究の方法

(1) 分散コードクローン分析

我々は、CCFinder と呼ぶコードクローン検出ツールを開発し、一般に配布してきており、この分野のデファクト・スタンダードツールになってきている。しかし、CCFinder は逐次的なサフィックス木アルゴリズムを用いて実現されており、対象とするプログラム全てを、符号化してメモリ上に展開する必要があるので、扱えるプログラムの大きさは約 500 万行が上限である。この問題を解決するために、対象を小さく分割して、多数の PC 上で分析し、結果を合成する手法を試みてきたが、より効率的で本格的な分散処理方式が望まれた。

①まず、サフィックス木アルゴリズムではなく、分散ハッシュを用いたアルゴリズムを用いて、多コア CPU 計算機や分散計算機上でより効率的にコードクローン分析ができる本質的な分散計算アルゴリズムを開発する。

②実際に、そのアルゴリズムを用いたシステムの開発を行い、分散計算機上でその評価を行う。評価対象として、オープンソースシステムや企業が開発したシステムを利用し、ソフトウェアの再利用状況の把握、予想外のシステム間の関連、ソフトウェアライセンスの不正利用などの用途に適用を行う。

③評価に基づき、システムの改良を行うとともに、外部での利用を考え、マニュアルやライセンスの整備を行う。

④完成した分散コードクローン分析システムをオープンソースシステムとして公開する。

(2) ソフトウェア生成物の大規模関連分析

我々は、ソフトウェアシステムの発展の歴史を、ソースコード集合間の類似関係を計算することにより視覚化する方法を研究してきており、例えば、BSD Unix OS のソースコードをクラスタ分類し、歴史的に近いシステムは、近いものとして分類できることを示した。また、ソフトウェア部品間の利用関係をグラフ化したものがいわゆるスケールフリーネットワークになることを発見し、この特性を利用したソフトウェア部品検索システム SPARS-J を開発、公開している。

本課題では、このようなソースコードの間の関連を主として、ソフトウェア生産物の間の関連をモデル化して分析し可視化する手法を開発するとともに、実用的なソフトウェア生産物関連閲覧システムの開発を行う。

①ソフトウェアシステムに含まれるソースコードの間で、特徴が類似している、より新しい、コードを一部共有している、などの種々の関連を定義し、それらに基づいて、ソフトウェアシステム群のモデルを構築するとともに、そのモデルを効率よく計算できる手法を開発する。このためには一部(a)の成果も利用する。

②開発した手法に基づいたプロトタイプシステムの開発を行い、例えば SourceForge 等のオープンソースシステムのレポジトリを対象にして分析し、評価を行う。

③プロトタイプシステムの評価を元にして、視覚化インターフェースを備えた実用システムの設計を行う。ここでは、多数の関連を効率よく分析するために分散処理を行うとともに、必要なシステムを容易に見つけ、快適に閲覧が行えるような手法を開発する。

④設計に基づいて、実用的なシステムの開発を行い、公開する。

(3) 大規模実証データ収集、分析

我々は、主として単一プロジェクトの実証データの収集、分析、フィードバックシステム EPM(Empirical Project Monitor) を開発し、多くの企業や大学に配布してきた。本課題では、EPM の考えを基礎とし、EPM のアーキテクチャを大幅に拡張して、多数のプロジェクトデータを効率よく処理し、プロジェクト間にまたがるデータ分析や予測ができるようなシステムの開発を行う。

①多数のプロジェクトから実証データを効率よく収集する方式の検討を行い、収集システムのプロトタイプシステムの開発を行う。

②収集した巨大な実証データに対して、プロジェクトの特性データの抽出、プロジェクト間の特性データの比較、そして、一部の特性データから他のプロジェクト特性の予測などを行う分析、予測手法の開発を行い、そのプロトタイプシステムの開発を行う。

③これらのシステムを統合するとともに、多コア計算機や分散計算機で効率よく稼動できるようにシステムを改良する。

④実際のプロジェクトで適用を行い、評価を行う。

(4) 計算ソフトウェア工学モデルの構築

上記(1)～(3)の処理で、頻繁に用いられるような計算を一般化して、分散処理システムで効率よく実行処理できるような、計算モデルを構築する。例えば、ソフトウェアプロジェクト、ソースファイル集合などを基本データとし、ソフトウェア集合の間の関連の計算、特徴データの抽出などを基本演算として、条件分岐や繰り返し、メタ演算などを揃えた計算モデルを開発し、それを分散計算システム上で効率よく実行する方式を開発する。

①分散計算を前提にした他の計算モデル（例えば Google で用いている MapReduce 等）及びソフトウェア工学技術の計算モデルを調査し、一般化に必要な基本演算や基本データの候補を選別する。また、他のサブテーマで用いられる基本演算や基本データを調査する。

②選んだ候補を元にして、他の制御構造を加えた計算モデルを開発し、論文等で発表を行う。

③計算モデルを分散システム上で効率よく実行する方式を検討し、プロトタイプシステムの構築を行う。
④種々のソフトウェア工学データに対して、プロトタイプシステムの実行を行い、効率よくデータの分析や予測が行え、プロジェクトの知見の有効利用が行えるかを評価する。

４．研究成果

本課題は，前述する４つの課題を有機的に結び付けた複数の問題として解決に取り組み，主に以下のような成果を得た．

(1) ソフトウェアの自動推薦システム A-SCORE

大量の部品から目的にあったソフトウェア部品を効率的に見つけるために，キーワード検索による部品検索システムが用いられている．部品検索システムとは，部品を機能や用途などで分類・管理して蓄積し，ユーザの指定に応じて部品を探し出すシステムである．多くの部品検索システムは，ユーザによって指定されたキーワード（検索語句）に適合する部品を検索する，キーワード検索機能を提供する．しかし，キーワード検索を用いた再利用には 2 つの問題点がある．第 1 に，開発者が意識しないと検索が行われないという問題である．第 2 に，開発者が適切なキーワードを選定する必要があるという問題である．これらの問題に対して，Ye らは，開発者が明示的に検索を行わなくてもシステムが自動的に部品を検索する，部品自動検索という概念を提案し，その手法を CodeBroker というツールとして実装した．部品自動検索では，開発者によるソースコードの編集を監視することで，開発者の手間を増やすことなく，自動的に再利用可能な部品を検索することができる．しかし，CodeBroker には，検索を行うタイミングが限られているという問題と，変更を加えずに再利用を行える部品しか検索できないというなどの問題が残されていた．

そこで本研究では，プログラム記述中の多くのタイミングで，変更が必要となるような部品をも自動的に推薦できる，新しい自動推薦手法を提案した．提案手法では，ソースコード中のコメントや識別子に基づいて検索を行う．また，検索の際には，自然言語に対する検索手法である潜在的意味インデキシング LSI（Latent Semantic Indexing）3）を応用して，曖昧さを許容して検索を行う．これにより，ある程度の差異がある部品も検索できるようになるため，変更を加えてから再利用する場合にも対応できるようになる．また，提案手法は，ドキュメントコメント以外にも様々な情報を利用して検索を行うため，メソッドのボディやクラス宣言など，ソースコードの様々な部分を記述しているときにも推薦を行うことができる．

また，提案手法を実装した部品自動推薦システム A-SCORE を作成し，学生を用いた実験によって A-SCORE による再利用の促進などの効果を評価した．実験では被験者にプログラミング課題を与え，再利用した既存部品の個数や完成したプログラムの不具合の数などを測定した．その結果，A-SCORE を利用したほうが再利用した部品の個数が多く，完成したプログラムの不具合も少なくなることが分かり，プログラムの品質が高くなることが明らかになった．また，既存手法では自動推薦が行えない場合において，自動推薦による再利用が行えた事例も存在したことが分かり，このことからも A-SCORE の有用性が示された．

(2) グラフマイニングアルゴリズムを用いたギャップを含むコードクローン情報生成

これまでに提案されているコードクローン検出手法のうち，Ueda ら手法 や Li らの手法など，一部の手法は Type3 のコードクローンを検出できるが，その他の多くの手法は Type3 のコードクローンを検出できない．しかし，コピーアンドペースト後に修正漏れが起こるという報告があり，そのようなコードクローンは Type3 になりえるため，タイプ 3 のコードクローンを検出することはソフトウェア保守の観点から重要である．検出するコードクローンの大きさの閾値を下げることにより，一つのギャップを含むコードクローンを複数の完全一致クローンまたは名前変更クローンとしては検出することは可能であるが，分析者自らがそれら複数のコードクローンを一つのギャップを含むコードクローンであると認識しなければならず，効率的に分析作業ができるとはいえない．

既存研究を踏まえ「ギャップを含むコードクローンをペアとしてだけでなく，セットとして検出可能な手法」「短いギャップだけでなく，大きなギャップに対しても，それらを含むコードクローンをある程度高速に検出可能な手法」の 2 つの特徴を持つ、ギャップを含むコードクローンの検出手法が必要であると考えた．

そこで本研究では、AGM アルゴリズムを用いてギャップを含むコードクローンを検出する手法を提案する．提案手法ではまず，既存の検出ツールを用いて，完全一致クローンと名前変更クローンを検出する．そして，それらを頂点とするグラフをソースファイル毎に構築し，そのグラフ中に存在する頻出パターンを抽出する．ギャップを含むコードクローンは，抽出された頻出パターンの各インスタンスであり，ペアではなく，セットとして検出される．また，グラフでは，大きなギャップも小さなギャップも 1 つの辺として

同様に扱われているため，ギャップの大きさに依存せずにコードクローンが検出される．しかし，グラフの集合から頻出パターンを抽出することは NP 完全として知られる部分グラフ同型同型であるため，膨大な計算コストを必要とする．この問題を改善するため，提案手法では，抽出するパターンに制限を加えることにより，少ない計算コストで検出処理を行った．

提案手法を検出ツール CCFinder のポストプロセッサとして実装し，どのようなギャップを含むコードクローン情報が生成されるのかを調査を行った．C 言語あるいは Java 言語で記述された四つのソフトウェアについて調査を行ったところ，生成結果の約 25%～35%が有益なコードクローン情報であることがわかった．

(3) 大規模ソフトウェアシステムを対象としたファイルクローンの検出

大規模なソースコード群からなるソフトウェアシステムではファイルの流用が頻繁に行われており、複数のプロジェクトによるコードクローンが大量に存在することが予想される。しかし、既存のコードクローン検出法はソースコードのミクロな部分に着目するものであり、大規模なソースコード群からなるソフトウェアシステム内にコードクローンがどの程度、どれくらいの規模で含まれているか、調査することはコストが高すぎて現実的ではない。また、発見した大量のコードクローンをコード単位で整理・統合すると多大なコストが必要となる。

そこで我々は、より調査コストの低いファイル単位でのクローン、ファイルクローンを高速に検出するツール FCFinder を開発した。ここでは、コメントを除外するなどの正規化を行ったときにソースコードが等しくなるファイルの関係をファイルクローンと呼び、それぞれのファイルをファイルクローンと呼ぶ。ファイルクローンは同値関係であり、同地類となるファイル集合をファイルクローンセットと呼ぶ。FCFinder は検出対象をファイルクローンに限定することによって、より高速にファイルクローンを検出できる。

本研究では、FCFinder を実装し、FreeBSD Ports Collection 全体に対して適用した。ソースコード 11.2GB 分という大規模な対象に対し、FCFinder は単一の計算機のみで 17 時間程度で解析を終了でき、既存システムよりも十分高速に、かつ小さい計算機資源でファイルクローンを検出できることがわかった。また、FreeBSD Ports Collection には多数のファイルクローンが含まれており、かつそのうち 1/4 程度はコメントやインデントだけが異なるものであることなど、既存ソフトウェア群に内在する多数のファイルクローン関係を明らかにすることができた。

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 7 件）

島田 隆次，市井 誠，早瀬 康裕，松下 誠，井上 克郎: "開発中のソースコードに基づくソフトウェア部品の自動推薦システム A-SCORE"，情報処理学会論文誌，Vol.50 No.12，pp. 3095-3107 (2009)．

肥後芳樹，宮崎宏海，楠本真二，井上克郎: "グラフマイニングアルゴリズムを用いたギャップを含むコードクローン情報の生成"，電子情報通信学会論文誌 D，vol. J93-D，no. 9，pages 1727-1735 (2010)．

肥後 芳樹，楠本 真二: "プログラム依存グラフを用いたコードクローン検出法の改善と評価，" 情報処理学会論文誌，volume 51，number 12，pages 2149-2168 (2010)．

堀田圭佑，佐野由希子，肥後芳樹，楠本真二，"修正頻度の比較に基づくソフトウェア修正作業量に 対する重複コードの影響に関する調査，" 情報処理学会論文誌，volume 52，number 9，pages 2788-2798 (2011)．

佐々木 裕介，山本 哲男，早瀬 康裕，井上克郎: "大規模ソフトウェアシステムを対象としたファイルクローンの検出"，電子情報通信学会論文誌 D Vol. J94-D No. 8 pp.1423-1433 (2011)．

Dotri Quoc, Kazuo Kobori, Norihiro Yoshida, Yoshiki Higo, Katsuro Inoue: "ModiChecker: Accessibility Excessiveness Analysis Tool for Java Program", Computer Software, Vol. 29, No.3, pp.212-218 (2012)．

Pei Xia, Makoto Matsushita, Norihiro Yoshida, Katsuro Inoue: "Studying Reuse of Out-dated Third-party Code in Open Source Projects", Computer Software, Vol.30, No.4, pp.98-104 (2013)．

〔学会発表〕（計 16 件）

Yuki Manabe, Yasuhiro Hayase, Katsuro Inoue: "Evolutional Analysis of Licenses in FOSS", Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), pp. 83-87 (2010)．

Daniel M. German, Yuki Manabe, Katsuro Inoue: "A Sentence-Matching Method for Automatic License Identification of Source Code Files", In Proceedings of the IEEE/ACM international Conference on Automated Software Engineering, pp. 437-446 (2010)．

Yusuke Sasaki, Tetsuo Yamamoto, Yasuhiro

Hayase, Katsuro Inoue: "Finding File Clones in FreeBSD Ports Collection", Proceedings of the 2010 7th IEEE Working Conference on Mining Software Repositories, pp.102-105 (2010).
Norihiro Yoshida, Takeshi Hattori, Katsuro Inoue: "Finding Similar Defects Using Synonymous Identifier Retrieval", Proceedings of the 4th International Workshop on Software Clones, pp.49-56 (2010).
Yui Sasaki, Keisuke Hotta, Yoshiki Higo, and Shinji Kusumoto, "Is Duplicate Code Good Or Bad? an Empirical Study with Multiple Investigation Methods and Multiple Detection Tools," In The 22nd annual International Symposium on Software Reliability Engineering (2011).
Yoshiki Higo, Yasushi Ueda, Minoru Nishino, and Shinji Kusumoto, "Incremental Code Clone Detection: a Pdg-Based Approach," In Proc. of 18th Working Conference on Reverse Engineering, pp.3-12 (2011).
Yu Kashima, Yasuhiro Hayase, Norihiro Yoshida, Yuki Manabe, Katsuro Inoue: "An investigation into the impact of software licenses on copy-and-paste reuse among OSS projects", Proc. of 18th Working Conference on Reverse Engineering, Limerick, Ireland, 2011, pp28-32 (2011).
Eunjong Choi, Norihiro Yoshida, Takashi Ishio, Katsuro Inoue, Tateki Sano: "Extracting Code Clones for Refactoring Using Combinations of Clone Metrics", Proceedings of the 5th International Workshop on Software Clones (IWSC 2011), pp.7-13 (2011).
Tomoko Kanemitsu, Yoshiki Higo, and Shinji Kusumoto, "A Visualization Method of Program Dependency Graph for Identifying Extract Method Opportunity," In Proc. of the 4th Workshop on Refactoring Tools (WRT2011), pages 8-14 (2011).
Keisuke Hotta, Yoshiki Higo, and Shinji Kusumoto, "Identifying, Tailoring, and Suggesting Form Template Method Refactoring Opportunities with Program Dependence Graph," In Proc. of the 16th European Conference on Software Maintenance and Reengineering (CSMR2012), pages 53-62 (2012).
Katsuro Inoue, Yusuke Sasaki, Pei Xia, Yuki Manabe: "Where Does This Code Come from and Where Does It Go? - Integrated Code History Tracker for Open Source Systems -", Proceedings of 34th ICSE, pp.331-341 (2012).
Yuki Yamanaka, Eunjong Choi, Norihiro Yoshida, Katsuro Inoue, Tateki Sano: "Industrial Application of Clone Change Management System", Proceedings of the 6th International Workshop on Software Clones (IWSC 2012), pp.67-71 (2012).
Eunjong Choi, Norihiro Yoshida, Katsuro Inoue: "What Kind of and How Clones are Refactored?: A Case Study of Three OSS Projects", Proceedings of the 5th Workshop on Refactoring Tools, pp.1-7 (2012).
Yui Sasaki, Tomoya Ishihara, Keisuke Hotta, Hideaki Hata, Yoshiki Higo, Hiroshi Igaki, and Shinji Kusumoto, "Preprocessing of Metrics Measurement Based on Simplifying Program Structures," In International Workshop on Software Analysis, Testing and Applications, pages 120-127 (2012).
Akira Goto, Norihiro Yoshida, Masakazu Ioka, Eunjong Choi, Katsuro Inoue: "Method Differentiator Using Slice-based Cohesion Metrics", AOSD'13, pp.11-14 (2013).
Yuki Yamanaka, Eunjong Choi, Norihiro Yoshida, Katsuro Inoue, Tateki Sano: "Applying Clone Change Notification System into an Industrial Development Process", Proceedings of the 21th ICPC, pp.199-206 (2013).

６．研究組織

(1)研究代表者

井上克郎（Katsuro Inoue）
大阪大学・大学院情報科学研究科・教授
研究者番号：20168438

(2)研究分担者

楠本真二（Shinji Kusumoto）
大阪大学・大学院情報科学研究科・教授
研究者番号：30234438
松下誠（Makoto Matsushita）
大阪大学・大学院情報科学研究科・准教授
研究者番号：60304028
石尾隆（Takashi Ishio）
大阪大学・大学院情報科学研究科・助教
研究者番号：60452413
岡野浩三（Kouzou Okano）
大阪大学・大学院情報科学研究科・准教授
研究者番号：70252632
肥後芳樹（Yoshiki Higo）
大阪大学・大学院情報科学研究科・助教
研究者番号：70452414