ソースコードの類似度評価基準作成とそれに基づいた類似度判定

渡　邉　卓　也†

著者は類似したプログラムを提示することでプログラミングを支援するツールの開発を行っている。そのツールにおいて類似度を判定する際に用いる計算式や検索に用いるアルゴリズムを改良する為には、検索結果の精度についての評価が欠かせない。現状では著者の主観的な評価に頼っているが、もし一般的な類似度についての基準が作成されれば、それを基に機械的な評価を行い、より客観的な精度についての議論が可能になるのではないかと期待している。

1. はじめに

著者は大学院生として類似プログラムを提示することによるプログラミング支援の研究に取り組んでおり、その実現の為のツールとして貯蔵庫から類似プログラムを検索して提示するツールを開発している[1)]。このツールの開発において、検索の際に用いるアルゴリズムや、ソースファイル同士の類似度を計算するのに用いる式の善し悪しを評価する為に検索結果の精度評価が必要であると考えている。しかしながら現状では著者個人の主観的な印象による評価に頼っている為、精度に関する主張は弱いものとならざるを得ない。そこで、様々な人々の幅広い意見を取り入れた、検索結果の精度評価に用いることが出来る程度に標準化された類似度の評価法を開発することが望ましいのではないかと考えてきた。

著者のソフトウェア開発経験は、趣味としてプログラミングを始めてからは二十年弱、しかし業務として開発に携わった経験は無い。Java を使い始めてからは十年弱であり、Java を研究用のプログラムの記述に用い始めてからは五年程度である。特に、先述のツールの一部は統合開発環境 Eclipse のプラグインである為 Java で記述されており、ツールを改良する為に日常的に Java でプログラムを記述している。

2. アンケート回答

2.1 ソース No.1 についての回答理由

観点 A については yes と回答したが、後の改変・拡張に備えて予め二つのクラスに分離したのであって今後明らかに異なるクラスに変化する場合等、一つのコードに纏めるのが適切でない場合も考え得る。

表 1 設問への回答

ソース No.	A	B	C	D	X
1	yes	yes	yes	yes	-
2	yes	yes	yes	yes	-
3	yes	yes	yes	yes	-
4	no	yes	yes	yes	-
5	no	yes	yes	yes	-
6	yes	yes	yes	yes	-
7	yes	yes	yes	yes	-
8	no	no	yes	no	-
9	no	yes	yes	yes	-
10	yes	yes	yes	yes	-
11	no	yes	yes	yes	-

2.2 ソース No.2 についての回答理由

1 と同様の理由で、観点 A については yes とはしたものの、必ずしも一つに纏めるのが適切とは言えない状況は考え得る。特に、メソッドの公開範囲の相違については後の拡張を想定した何らかの明確な意図に基づく違いである可能性を考えておくべきではなかろうか。

2.3 ソース No.3 についての回答理由

観点 A については、一つのコードに纏めるよりも差分を子クラスとして分離する方が適切な場合も考え得る。

2.4 ソース No.4 についての回答理由

共通の親クラスに纏めて記述出来るであろう箇所は存在するが、扱うデータの種類に応じて二つのクラスに分離すること自体は適切と感じたので、観点 A については no と回答した。

2.5 ソース No.5 についての回答理由

TIntArrayList の実装しているインタフェースによっては型変数を用いて一つのクラスに纏めることが可能と思われるが、ここでは纏めることが困難な場合を考

† 東京大学
The University of Tokyo

え、観点 A について no と回答した。

2.6 ソース No.6 についての回答理由

コード片 6-1 でインポートしている AccessibleState クラスを利用していないのは不自然に感じられるが、少なくとも形式上は型変数を用いて一つのクラスに纏めることは可能なので、観点 A につき yes と回答した。

2.8 ソース No.8 についての回答理由

観点 C については、sort() メソッドの一部分を再利用する場合には一方のコードを使用可能な場所で他方のコードを使用することは難しいと思われるが、sort() メソッド全体は互いに互換と考えられるので yes と回答した。

3. 議　　論

著者は第 1 章で述べたように類似したソースコードをプログラマに提示することでプログラミングを支援するツールを開発している。このツールは統合開発環境上でのプログラマの入力を監視し、プログラマがエディタで編集中のプログラム全体を適切なタイミングで読み込み、そのプログラムと貯蔵庫中に保存されているプログラムとの類似度を計算する。その結果類似度が高いと判定された貯蔵庫中のプログラムの幾つかについて、中でも最も類似度が高いと考えられる部分を統合開発環境上で表示することでプログラマの参考となることを目指している。

このツールの開発においては、ソースコード同士の類似度を算出する際に用いる計算式や、ソースコード中の最も類似している部分を判定する為のアルゴリズムが様々考えられ、従ってそれらの優劣を比較検討する必要がある。しかし、こと検索結果の精度に関してはまず著者本人の主観による評価に頼らざるを得ず、その評価の妥当性には確信が持てない。最終的には被験者を募って実験を行い、被験者による主観評価を平均することである程度の客観性を確保出来るとはいえ、細かなパラメータの調整作業の過程でそのような実験を毎回行うのは非現実的であり、何らかの機械的な精度の判定手法が存在すると大いに研究に資するものと考えられる。また、そのような判定手法は、他の研究で用いられている検索手法と自分の検索手法を比較する際においてもより客観的な評価基準を提供してくれるものと期待される。

そうした判定手法を開発するにあたっては、様々な研究者や開発実務担当者の意見を総合し、幅広く受け入れられる類似度の基準を作成することが欠かせない。また、その基準の利用目的により、「類似度」の内実が異なってくることも考えられる。例えば、コード片 9-1 と 9-2 はほぼ同じ目的のフィールドやメソッドで構成されており、片方を改変する際にはもう片方も改変せねばならない可能性が高いが、一方で、型に着目し、byte 型の使われ方を知りたいといったような場合には 9-1 やその他の byte 型を多く用いているプログラムが類似物として抽出され、逆に 9-2 は類似プログラムとは判定されないのが望ましい。従って、様々な目的に応じた幾つかのなるべく直交した複数の観点から類似度を構成することが望ましい。

そうした基準を作成した上で（或は同時進行的に）、機械的評価の為の標準的な評価集合を作成したい。この評価集合は世の中に存在するソースコードを代表するものでなければならず、よって様々な種類のソースコードから満遍なく抽出され、かつ類似度判定を行うに十分な数のソースコードから構成されている必要がある。

上記、

- 標準化された類似度の判定基準
- 代表性を有する、評価の為のソースコード集合

の二つを得た上で、類似度の機械的評価の為には評価集合中のソースコード同士の類似度を予め求めておく必要がある。この予め求めておいた類似度と、評価対象の類似プログラム検索手法を評価集合に対して適用した結果とを比較し、精度を求めるのである。類似度の算出は人手によることも考えられるが、判定基準に基づいてある程度自動的に行うことが出来る部分については自動的に行うのが望ましいであろう。

このように精度の判定過程が標準化され、関連した研究につき共通の基準で評価することが可能となることは、多くの研究者にとって利益となるものと思われる。著者は本ワークショップがそのような標準化の第一歩となるものと期待している。

謝辞　このような興味深い問題につきワークショップを組織され、また紹介下さった大阪大学の石尾隆氏に深く感謝いたします。

参　考　文　献

1) 渡邉卓也, 増原英彦, 2007,「類似プログラムの提示ツール Selene」, 日本ソフトウェア科学会第 24 回大会講演論文集, 3B-2