Form 3

Abstract of Thesis

Name (楊 嘉晨)	
Title	Structural and Semantic Code Analysis for Program Comprehension (構造的及び意味的側面からのプログラム理解支援)

Abstract of Thesis

It is important to understand the software during its development and maintenance. Understanding the internal mechanics of a system implies studying its documentation and source code during a given maintenance task. There are two dimensions of information that can be obtained from static analyses to aid program comprehension, namely structural information and semantic information. Structural information refers to issues such as the actual syntactic structure of the program along with the control and data flow that it represents, The other dimension, semantic information, refers to the domain specific issues of a software system.

This dissertation addresses these two dimensions of program comprehension with two studies. The first one is a study on a classification model of source code clones, which helps understanding the source code from structural similarity. The latter one focuses on an important aspect of semantic information, which is the purity of the code modules.

The first study focuses on code clones, which can represent the structural similarity of programs. A code clone, or simply a clone, is defined as 'a code fragment that has identical or similar code fragments to one another'. The presence of code clones is pointed out as a bad smell for software maintenance. The reason is that: if we need to make a change in one place, we will probably need to change the others as well, but we sometimes might miss it. According to a survey we conducted, users of code detectors tend to classify clones differently based on each user's individual uses, purposes or experiences about code clones. A true code clone for one user may be a false code clone for another user. From this observation, we propose an idea of studying the judgments of each user regarding clones. The proposed technique is a new clone classification method, entitled Filter for Individual user on code Clone Analysis (Fica). Code clones are classified by Fica according to a comparison of their token type sequences through their similarity, based on the "term frequency - inverse document frequency" (TF-IDF) vector. Fica learns user opinions concerning these code clones from the classification results. In a production environment, adapting the method described in this research will decrease the time that a user spends on analyzing code clones.

The second study focuses on understanding the purity and side effects of a given program. It is difficult for programmers to reuse software components without fully understanding their behavior. The documentation and naming of these components usually focus on intent, i.e., what these functions are required to do, but fails to illustrate their side effects, i.e., how these functions accomplish their tasks. It is hard to understand and reuse modularized components, because of the possible side effects in API libraries. In addition, undocumented API side effects may be changed during software maintenance, making debugging even more challenging in the future. By understanding side effects in the software libraries, programmers can perform high level refactoring on the source code that is using the functional part of the libraries. Moreover, the calculation without side effects are good candidates for parallelization. However, the purity information is usually missing in external libraries, therefore programmers would risk introducing bugs with such refactorings. In this study, we present an approach to infer a function's purity from byte code for later use. Programmers can use effect information to understand a function's side effects in order to reuse it. Furthermore, we conducted a case study on purity guided refactoring based on gathered purity information. As a case study, we applied a kind of the purity-guided refactoring, namely Memoization refactoring on several open-source libraries in Java. We observed improvements of their performance and preservation of their semantics by profiling the bundled test cases of these libraries.

様式７

論文審査の結果の要旨及び担当者

氏　名　（楊　嘉晨　）			
	（職）		氏　名
論文審査担当者	主 査	教授	楠本真二
	副 査	教授	井上克郎
	副 査	教授	増澤利光
	副 査	准教授	肥後芳樹

論文審査の結果の要旨

本論文は、ソフトウェア開発や保守において重要な、不具合修正や機能変更等で、最初に行われるプログラム理解の活動を支援することを目的としており、特に静的なソースコード分析方法に対する新たな提案を行っている。静的なソースコード分析とは、プログラムを実行せずに様々な性質を調べ、開発者に有用な情報を提示することで、プログラム理解の効率化を支援する技術である。本論文では、静的なソースコード分析にあたって、構造的な分析と意味的な分析の２つのアプローチをとっている。一つは、構造的なコードクローン分析手法であり、もう一方は意味的なコード純粋性分析手法である。

構造的なコードクローン分析手法として、機械学習を用いたコードクローン分析手法を提案し、その手法をツールとして開発している。コードクローンとはソースコード中に存在する同一、あるいは、類似するコード片のことを意味し、主にコード片のコピーアンドペーストにより生成される。近年、コードクローンはソフトウェア工学、特に、ソフトウェア保守の研究分野において注目を集めている研究テーマである。また、コードクローンが原因となり失敗した大規模ソフトウェア開発プロジェクトの事例も報告されており、実用面においても重要なテーマとなっている。一般に同じコードクローンであっても、開発者のコンテキストによって意味のあるコードクローンであるかどうかは異なることが指摘されている。提案手法では、開発者に予め与えたコードクローンの評価結果に基づいて機械学習を適用することで、別のコードクローンがその開発者にとって意味があるかどうかを識別する。提案手法の評価にあたって、開発したツールをインターネット上で公開し、３２人の被験者が参加した大規模なコードクローン分析実験を行っている。実験の結果、提案手法は既存研究に比べてコードクローンの識別精度が高く、実際の開発現場における有用性も高いことが確認されている。

次に、オブジェクト指向言語であるJavaで記述されたソースコードに対して、意味的なコード純粋性分析手法を提案し、コードから純粋な関数や副作用を持つ関数を自動検出するツールを開発している。純粋性とは評価対象の関数が他の関数に対し、他の状態に依存せずに影響も与えないことを保証するものであり、純粋な関数は理解しやすく、並列化も容易にできるという特性がある。純粋性は本来関数型言語に対して定義されていた概念であるが、本論文ではオブジェクト指向言語に対して適用できるよう定義している。それにより、近年のマルチコアCPUにおいて、既存の逐次プログラムを並列化することによる高速化への要望にも容易に対応できる。提案手法による関数の純粋性評価精度と並列化の可能性を複数のオープンソース・ソフトウェアに適用した結果、純粋性な関数、及び、副作用を持つ関数を自動的に分類でき、純粋性に関係する不具合を容易に検出できている。更に、応用例として、メモ化リファクタリングをオープンソース・ソフトウェアのライブラリに適用し、性能向上が確認されている。これらの結果は、提案手法のプログラム理解支援やプログラム自動並列化に対する実用性が期待できるものである。

以上のように、本論文の成果は、プログラム理解を支援する手法として、技術面、並びに、実用面において高い貢献があると考えられ、博士（情報科学）の学位論文として価値のあるものと認める。