"УТВЕРЖДАЮ"

Директор ФИЦ ИУ РАН
академик РАН И.А.Соколов

«26» февраля 2016 г.

ОТЗЫВ ВЕДУЩЕЙ ОРГАНИЗАЦИИ

на диссертационную работу Саргсяна Севака Сениковича «Методы поиска клонов кода и семантических ошибок на основе семантического анализа программы», представленную к защите на соискание ученой степени кандидата физико-математических наук по специальности 05.13.11– «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»

Актуальность. Одним из этапов безопасной разработки программного обеспечения (ПО) является автоматическая проверка качества исходного кода. В ходе проверки выявляются различные ошибки, допущенные программистом, что позволяет снизить затраты на поиск ошибок и поддержку ПО. Часто причиной возникновения ошибок является копирование исходного кода и использование интернет ресурсов для поиска подходящего кода с последующей его адаптацией. Для автоматического анализа качества исходного кода используются инструменты поиска клонов кода и поиска ошибок в скопированных и некорректно адаптированных фрагментах кода. Существующие инструменты поиска клонов кода либо не находят все клоны кода, либо не масштабируются для анализа больших проектов с десятками миллионов строк исходного кода. Инструменты поиска ошибок в скопированных и неверно адаптированных фрагментах кода либо не умеют находить все допущенные ошибки, либо выдают слишком

много ложных срабатываний. Таким образом, тема диссертации С.С. Саргсяна, посвященная методам создания высокоточных и масштабируемых инструментов поиска клонов кода и ошибок, возникающих из-за некорректно адаптированных вставленных фрагментов кода, является актуальной.

Основные результаты. В диссертации рассматривается метод поиска клонов на основе семантического анализа программы, состоящий из четырех фаз: (1) разделение графа зависимостей программы (ГЗП) на подграфы; (2) фильтрация несхожих пар ГЗП с применением линейных алгоритмов; (3) поиск схожих подграфов максимального размера в паре ГЗП; (4) фильтрация ложных срабатываний. Разделение на четыре фазы позволяет обеспечить масштабируемость инструмента на десятки миллионов строк исходного кода и обеспечить высокое качество работы.

К основным результатам диссертации можно отнести следующие.

1) Предложенный автором новый алгоритм разделения ГЗП на подграфы позволяет получать подграфы такие, что потенциальные клоны кода полностью оказываются в одном подграфе. Размеры подграфов приблизительно одинаковы. Сравнительное тестирование разработанного алгоритма на наборе тестов (Linux 2.6, Firefox Mozilla, LLVM/Clang и OpenSSL) показало, что предложенный алгоритм позволяет находить в среднем 1.5-2 раза больше клонов кода.
2) Применение алгоритмов с линейной сложностью для исключения несхожих пар ГЗП позволяет обрабатывать большинство пар ГЗП за линейное время. Тяжеловесные алгоритмы поиска схожих подграфов запускаются только для небольшого количества пар ГЗП. Такой подход позволяет ускорить работу инструмента в несколько раз.
3) Автором предложены три новых приближенных алгоритма для поиска схожих подграфов максимального размера:
   - алгоритм на основе слайсинга;
   - преобразование ГЗП в дерево с последующим поиском изоморфных поддеревьев максимального размера;
   - алгоритм на основе метрики для вершин ГЗП.

   Алгоритм на основе слайсинга способен обнаружить все клоны кода, но работает дольше остальных. Алгоритм на основе деревьев имеет высокую точность при поиске

скопированных фрагментов кода, в которых были произведены только замены идентификаторов. Алгоритм использующий метрику имеет сравнительно низкую точность, но работает быстрее всех.

4) После того, как все пары схожих подграфов найдены, производится дополнительная проверка соответствующего исходного кода, что позволяет отфильтровать большинство ложных срабатываний.

В диссертации приводится архитектура инструмента поиска клонов кода. Инструмент состоит из трех основных частей. Первая часть отвечает за генерацию ГЗП и реализована как проход компиляторной инфраструктуры LLVM. Во время компиляции проекта генерируются и сохраняются ГЗП. Инструмент имеет возможность генерировать три варианта ГЗП, что позволяет эффективно решать конкретную задачу. Вторая часть отвечает за поиск клонов кода и реализована как инструмент компиляторной инфраструктуры LLVM. Последняя часть отвечает за автоматическую генерацию клонов кода и проверку точности реализованных алгоритмов. Для генерации клонов кода используются стандартные и обфусцирующие проходы LLVM. Автором проведен анализ точности реализованных алгоритмов на проектах Linux 2.6, Firefox Mozilla, LLVM/Clang и OpenSSL. Согласно результатам, реализованный инструмент обладает точностью более 90 процентов.

Инструмент был применен для анализа проектов: Linux 2.6 (содержит приблизительно 14 миллионов строк исходного кода), Firefox Mozilla, LLVM/Clang и OpenSSL. Для ядра Linux-2.6 было найдено около 2000 клонов кода с длинной более 25 строк исходного кода. На синтетическом наборе тестов было проведено сравнение точности реализованного инструмента с тремя известными инструментами поиска клонов кода: MOSS, CCFinder и CloneDR. Результаты показали, что реализованный инструмент обладает большей точностью.

Изменения, сделанные в динамическом компиляторе V8 языка JavaScript, позволили применить реализованный инструмент поиска клонов кода для языка JavaScript. На основе промежуточного представления Hydrogen компилятора V8 генерируется ГЗП. Автором были проанализированы и выявлены клоны кода в тестовых наборах SunSpider, Octane и Kraken.

В диссертации представлен метод поиска семантических ошибок, возникающих при некорректной адаптации скопированных фрагментов кода к контексту, в который они перенесены. Предложенный автором метод комбинированно использует лексический и семантический анализ, что позволяет находить семантические ошибки с большой точностью. Метод состоит из двух основных этапов. На первом этапе производится поиск всех скопированных фрагментов кода, которые различаются только идентификаторами. На втором этапе строится ГЗП скопированных фрагментов кода. Дальнейший анализ построенных ГЗП позволяет определить допущенную ошибку. Автором были проанализированы и выявлены более сотни ошибок в проектах Android 4.3, Linux 2.6, Firefox Mozilla, LLVM/Clang, OpenSSL и Qemu.

Новизна полученных результатов заключается в том, что разработаны:

- архитектура инструмента поиска клонов кода, который легко расширяется для новых языков программирования;
- четырехфазный метод поиска клонов кода, обеспечивающий поиск клонов кода с высокой точностью и масштабируемость инструмента; подсистема анализа точности реализованных алгоритмов;
- алгоритм разделения ГЗП на подграфы; алгоритмы поиска клонов кода на основе слайсинга, метрики и изоморфизма деревьев;
- комбинированный метод поиска семантических ошибок базирующейся на лексическом и семантическом анализе программы.

Подводя итог, можно сделать вывод о том, что автором проведен большой объем исследований в области поиска клонов кода и связанных с ними семантических ошибок в исходном коде программы. Приведенные алгоритмы показали свою эффективность в ходе тестирования на ряде больших проектах с открытым исходным кодом: Android 4.3, Linux 2.6, Firefox Mozilla, LLVM/Clang ,OpenSSL и Qemu. Создана специальная подсистема для анализа точности реализованных алгоритмов поиска клонов кода, показавшая, что предложенные алгоритмы обладают высокой точностью (более 90%). Произведено сравнение реализованных алгоритмов с тремя известными инструментами поиска клонов кода: MOSS, CCFinder и CloneDR. Согласно результатам, предложенные алгоритмы обладают большей точностью.

Разработанные автором методы и инструменты поиска клонов кода и семантических ошибок позволяют провести качественный систематический анализ больших проектов, что является важной задачей с точки зрения науки и практики. Разработанные инструменты могут быть использованы в цикле разработки ПО, что позволит выявлять допущенные ошибки во время разработки. Это позволит снизить затраты на разработку и дальнейшую поддержку ПО.

Замечания.

1. В разделе 3.1.1 рассматривается возможность генерации ГЗП тремя разными способами в зависимости от задачи, но не производится оценка замедления инструмента каждом случае.
2. В главе 4 не объясняется, как строится граф зависимостей программы для не типизированного языка JavaScript. В частности, возникает вопрос, как определяются типы переменных в этом случае.

Реализованные инструменты внедрены в Институте системного программирования Российской академии наук (ИСП РАН) и используются в цикле разработки ПО.

Диссертационная работа Саргсяна Севака Сениковича соответствует всем требованиям ВАК РФ, предъявляемым к диссертациям на соискание ученой степени кандидата физико-математических наук по специальности 05.13.11 - «математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей», Саргсян Севак Сеникович заслуживает присуждения ему искомой степени.

Отзыв обсужден и одобрен25 февраля 2016 годана семинаре отдела Систем Математического Обеспечения Вычислительного центра им. А.А.Дородницына Российской академии наук Федерального исследовательского центра «Информатика и управление» Российской академии наук, протокол № 02/02-2016

Зав. отделом Систем Математического
Обеспечения Вычислительного
Центра ФИЦ ИУ РАН
Д.ф.-м.н. В.А.Серебряков