УДК 004.054, 004.4'233

И.В. РУДАКОВ, В.О. МЕДВЕДЕВ
ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)», г. Москва

АЛГОРИТМ ВЕРИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ПОМОЩЬЮ ИЕРАРХИЧЕСКИХ СЕТЕЙ ПЕТРИ

Ключевые слова: верификация; статический анализ кода; статическое тестирование; иерархическая сеть Петри; исполнимая модель; зацикливание программного обеспечения.

Аннотация: Целью данной работы является разработка алгоритмов, позволяющих осуществлять проверку программного обеспечения на наличие зацикливаний с помощью модели потоков данных, формализованной иерархической сетью Петри, построенной в результате анализа объектно-ориентированного кода. Зацикливанием в рамках этой работы будем считать такое состояние программного обеспечения, при котором осуществляется замкнутый цикл вызовов методов классов. Авторами была выдвинута гипотеза, согласно которой иерархическая сеть Петри может являться формализацией потока данных при вызове методов и событий. В рамках данной работы были разработаны алгоритм преобразования исходных текстов программного обеспечения в иерархическую сеть Петри и алгоритм проверки иерархической сети Петри на наличие зацикливаний. Решение этих задач позволило осуществлять проверку исходных текстов программного обеспечения на наличие зацикливаний без привязки к языку программирования, что особенно важно с учетом темпов развития инструментов проектирования программного обеспечения. В результате опытным путем было установлено, что достоверность анализа составила чуть более 86 %.

В настоящее время наиболее популярным подходом к созданию программных продуктов является объектно-ориентированный подход, поскольку он позволяет наиболее просто для восприятия человеком описывать сложные структуры [1]. Данный подход позволяет осуществлять разработку более сложных программных продуктов за счет возможности создания иерархии объектов. Однако увеличение сложности стало приводить и к увеличению ошибок, допускаемых на различных этапах разработки программного обеспечения, при этом затрудняя их быстрое обнаружение.

Для обеспечения корректности и надежности работы информационных систем используют различные методы верификации и валидации, позволяющие выявлять ошибки на разных этапах разработки и сопровождения программного обеспечения. На основании этого широкое распространение получили средства верификации – статические анализаторы исходных текстов программного обеспечения (**ПО**), которые позволяют выявлять дефекты уже на этапе конструирования.

Одним из перспективных инструментов представления модели потоков данных является иерархическая сеть Петри [2]. Параллелизм иерархических сетей Петри позволяет явно учитывать перегрузку функций и операторов, наследование классов, использование интерфейсов и т.п. Верификация модели потоков данных позволяет выявлять дефекты программного обеспечения не только внутри модуля, но и между модулями.

Авторами статьи был разработан алгоритм поиска зацикливаний ПО с помощью верификации модели потоков данных, формализованной иерархической сетью Петри, построенной в результате анализа объектно-ориентированного кода. Зацикливанием в рамках этой работы будем считать такое состояние ПО, при котором осуществляется замкнутый цикл вызовов методов классов, т.е. выход из подобной «петли» невозможен. Подобные дефекты разработанные на сегодняшний день статические анализаторы кода не обнаруживают. При этом динамические

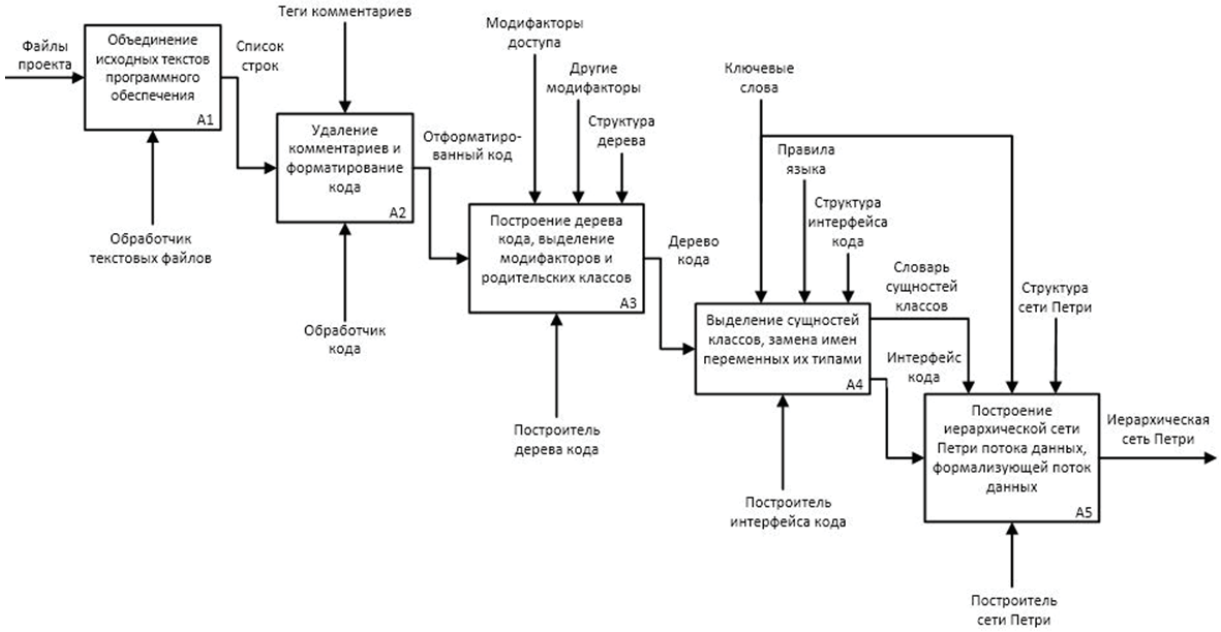

Рис. 1. *IDEF*0-диаграмма преобразования исходных текстов ПО в диаграмму потоков данных, формализованную иерархической сетью Петри

анализаторы кода ввиду особенностей своей работы не всегда могут обнаружить зацикливания, поскольку бывают ситуации, когда причиной зацикливания является редко вызываемый участок кода.

На сегодняшний день существует несколько десятков языков программирования, позволяющих осуществлять разработку с использованием объектно-ориентированного подхода. Поскольку каждый язык программирования имеет свои синтаксические особенности, то для построения иерархической сети Петри необходима предварительная обработка исходных текстов анализируемого программного обеспечения.

Все объектно-ориентированные языки программирования имеют следующие сущности [3].

1. Класс – тип данных, описывающий структуру и поведение объектов.
2. Объект – экземпляр класса.
3. Поле (атрибут) – элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса.
4. Метод – процедура или функция, выполняющаяся в контексте объекта, для которого она вызывается.
5. Модификатор – ключевое слово, определяющее свойство описываемой сущности (класса, метода, атрибута и т.п.). Выделяют модификаторы доступа и «другие» модификаторы, обычно описывающие вид сущности (абстрактный, статический, асинхронный и т.п.).

Также некоторые объектно-ориентированные языки программирования оперируют понятием «пространство имен». Пространства имен позволяют разделить сущности на области видимости.

Таким образом, для универсального преобразования исходных текстов программного обеспечения в иерархическую сеть Петри необходимо оперировать только вышеуказанными сущностями. Этапы преобразования исходных текстов программного обеспечения в диаграмму потоков данных, формализованную иерархической сетью Петри, представлены на рис. 1.

На первом этапе объединяется исходный код проекта, который может содержаться в нескольких файлах. Далее удаляются все комментарии, удаляется форматирование. Список строк объединяется в одну строку.

На третьем этапе выделяются уровни иерархии – пространства имен (*namespace*), классы (*class*) и наполнение класса. Формируется дерево кода. На этом этапе происходит обработка на уровне классов – у них выделяются модификаторы доступа, другие модификаторы и родительские (базовые) классы.

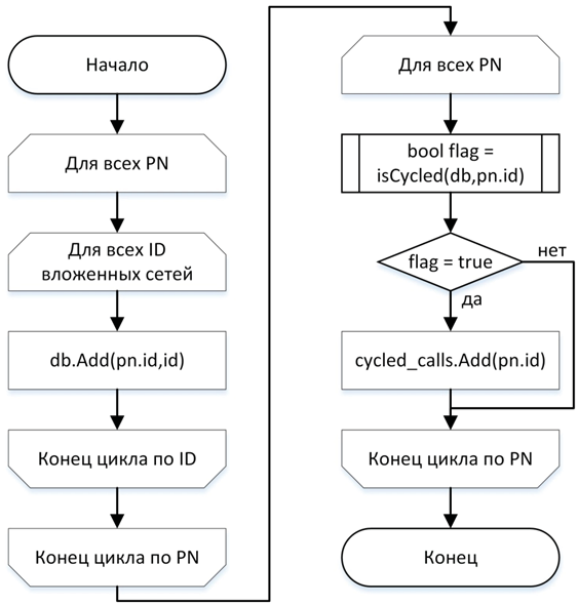

Рис. 2. Общая схема алгоритма проверки иерархической сети Петри на наличие зацикливаний

На следующем этапе происходит выделение сущностей класса, модификаторов, типов, входных параметры и ряд других признаков. В коде сущностей происходит замена имен переменных и констант их типами согласно области видимости. В результате формируется интерфейс кода, а также словарь сущностей классов.

На заключительном этапе строится иерархическая сеть Петри. Вызов методов или событий – это вложенные сети Петри. В результате полученная иерархическая сеть Петри – это формализация потока данных при вызове методов и событий. Структура иерархической сети Петри содержит идентификатор сети, список позиций, список переходов, список вложенных сетей и список связей [4].

Для поиска зацикливаний был разработан алгоритм, общая схема которого представлена на рис. 2. В основе алгоритма лежат два прохода по вложенным сетям иерархических сетей Петри. После первого прохода формируется словарь вызовов, представляющий собой список пар «откуда вызываем – что вызываем». Во время второго прохода, если выясняется, что имеется зацикливание на текущую сеть Петри, идентификатор сети Петри добавляется в словарь сетей, имеющих зацикливание.

В данной схеме алгоритма имеется вызов функции *isCycled*(), которая проверяет сеть Петри на наличие зацикливаний. Проверка завершается в случае, если обнаружено зацикливание, или в случае, если больше не осталось таких сетей Петри, которые не были обработаны.

Разработанные алгоритмы были реализованы в системе статического анализатора для языка *C#*. В результате анализа 100 проектов был получен список из 108 методов, вызов которых, по мнению анализатора, приводит к зацикливанию. Ложными из них оказались 15 методов, что составляет 13,89 % от общего количества срабатываний. При этом стоит отметить, что в общей сложности во всех проектах 40 254 вызова методов классов исходного кода.

Таким образом, полученная опытным путем достоверность анализа составила приблизительно 86 %, что является достаточно высоким показателем для статических анализаторов кода.

Список литературы

1. Вайсфельд, М. Объектно-ориентированное мышление / М. Вайсфельд. – СПБ. : Питер, 2014. – 304 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5-496-00793-1.
2. Медведев, В.О. Верификация программного обеспечения, формализованного сетью Петри / В.О. Медведев, И.В. Рудаков // Современные научные исследования и разработки. – 2017. – № 7(15). – С. 230–233.
3. Хорев, П.Б. Объектно-ориентированное программирование с примерами на C# / П.Б. Хорев. – М. : Форум, НИЦ ИНФРА-М, 2016. – 200 с.: ил. – (Высшее образование: Бакалавриат). ISBN 978-5-00091-144-0.
4. Коротиков, С.В. Применение сетей Петри в разработке программного обеспечения центров дистанционного контроля и управления: дисс. ... канд. тех. наук / С.В. Коротиков. – Новосибирск : НГТУ, 2007.

References

1. Vajsfel'd, M. Ob#ektno-orientirovannoe myshlenie / M. Vajsfel'd. – SPB. : Piter, 2014. – 304 s.:

il. – (Serija «Biblioteka programmista»). ISBN 978-5-496-00793-1.

2. Medvedev, V.O. Verifikacija programmnogo obespechenija, formalizovannogo set'ju Petri / V.O. Medvedev, I.V. Rudakov // Sovremennye nauchnye issledovanija i razrabotki. – 2017. – № 7(15). – S. 230–233.

3. Horev, P.B. Ob#ektno-orientirovannoe programmirovanie s primerami na C# / P.B. Horev. – M. : Forum, NIC INFRA-M, 2016. – 200 s.: il. – (Vysshee obrazovanie: Bakalavriat). ISBN 978-5-00091-144-0.

4. Korotikov, S.V. Primenenie setej Petri v razrabotke programmnogo obespechenija centrov distancionnogo kontrolja i upravlenija: diss. ... kand. teh. nauk / S.V. Korotikov. – Novosibirsk : NGTU, 2007.

I.V. Rudakov, V.O. Medvedev
Bauman Moscow State Technical University, Moscow

An Algorithm of Software Verification Using Hierarchical Petri Nets

Keywords: verification; static code analysis; static testing; hierarchical Petri net; executable model; software looping.

Abstract: In this paper, we describe algorithms, which allow checking software for the presence of looping using a data flow model formalized by a hierarchical Petri net, which has been built as a result of the object-oriented source code analysis. Looping in this work is considered a software state in which a closed cycle of class methods calls is performed. The authors hypothesized that the hierarchical Petri net may be a formalization of the data flow when calling methods and events. An algorithm for transforming software source texts into a hierarchical Petri net and an algorithm for checking the hierarchical Petri net for looping was developed. The developed algorithms allow checking the source code of the software for the presence of looping without binding to the concrete programming language, which is especially important taking into account the pace of development of software design tools. As a result, it was established experimentally that the reliability of the analysis was just over 86 %.
