УДК 004.4′2
doi:10.15217/issn1684-8853.2018.2.67

ОБЗОР ДИНАМИЧЕСКИХ МЕТОДОВ ВОССТАНОВЛЕНИЯ ЧАСТИЧНЫХ СПЕЦИФИКАЦИЙ ПРОГРАММНЫХ БИБЛИОТЕК НА ОСНОВЕ АНАЛИЗА ПРОГРАММНЫХ ПРОЕКТОВ

И. С. Егорова[a], *аспирант, is.egorova@mail.ru*
В. М. Ицыксон[a], *канд. техн. наук, доцент, vlad@icc.spbstu.ru*
[a]*Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург, Политехническая ул., 29, 195251, РФ*

Введение: *в связи с тем, что значительная часть программного обеспечения в настоящее время разрабатывается с использованием сторонних библиотек и фреймворков, которые обычно плохо документированы, становится актуальной задача воссоздания соответствующей документации.* ***Цель:*** *поиск перспективных подходов автоматизированного извлечения частичных формальных спецификаций программных библиотек, основанных на методах динамического анализа программных проектов, использующих эти библиотеки.* ***Результаты:*** *произведены обзор, сравнение и классификация различных формализмов, используемых для описания программных компонентов. Осуществлен обзор подходов, которые могут быть использованы для извлечения такого описания методами динамического анализа. Установлено, что наибольшей точностью из рассматриваемых методов динамического анализа вне зависимости от вида фиксируемых свойств обладают методы, основанные на генерации и проверке гипотез на основе шаблонов. К основным ограничениям подходов группы относятся невозможность восстановления с их использованием спецификаций, описывающих взаимодействие нескольких объектов и изменение состояния библиотеки, а также использование шаблонов. Помимо этого, необходимо отметить, что на данный момент ни один из рассмотренных подходов, основанных на методах как статического, так и динамического анализа, не поддерживает извлечение семантического описания вызовов библиотечных компонентов в процессе восстановления спецификации.*

Ключевые слова *— спецификация программной библиотеки, формальная спецификация, динамическое извлечение спецификаций, расширенные конечные автоматы, временные свойства, поведенческое описание библиотеки, комплексные правила, динамический анализ.*

Цитирование: Егорова И. С., Ицыксон В. М. Обзор динамических методов восстановления частичных спецификаций программных библиотек на основе анализа программных проектов // Информационно-управляющие системы. 2018. № 2. С. 67–75. doi:10.15217/issn1684-8853.2018.2.67
Citation: Egorova I. S., Itsykson V. M. Review of Dynamic Methods for Extraction of Partial Software Library Specification. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2018, no. 2, pp. 67–75 (In Russian). doi:10.15217/issn1684-8853.2018.2.67

Введение

Данная работа является продолжением статьи [1], в которой были представлены результаты обзора формализмов, используемых для восстановления спецификаций компонентов программных библиотек путем анализа проектов, реализованных с их применением, а также результаты обзора подходов для извлечения таких спецификаций с применением методов статического анализа. Объектом изучения, представленного в ней, так же, как и в работе [1], являются подходы, применяемые для автоматизированного восстановления спецификаций программных библиотек путем использования методов эмпирической программной инженерии и извлечения данных из репозиториев. Предметом настоящего обзора выступает исследование подходов, основанных на методах динамического анализа исходного кода программных проектов с точки зрения возможности восстановления с их помощью наиболее развернутого описания правил использования библиотечных компонентов и достижения наилучшего качества извлекаемых спецификаций. При этом конечной целью является выбор подхода или группы подходов, позволяющих формировать спецификацию анализируемой библиотеки в терминах формализма [2], разработанного ранее авторами.

К отличительным особенностям формализма [2] относятся поддержка возможности отображения в виде иерархии конечных автоматов (КА) комплексных правил, характеризующих как объекты библиотечных классов, так и связи, существующие между ними и состоянием самой библиотеки, а также возможность задания семантики вызовов методов библиотеки.

Среди изученных подходов, основанных на анализе исходного кода программ, использующих библиотеки, извлекать подмножество более простых комплексных правил позволяет только подход, представленный в работе [3], однако информации о систематической оценке качества результирующих спецификаций, получаемых

с его применением, не приводится. Наилучших результатов с точки зрения выразительности и качества (точности порядка 60 %) извлекаемых спецификаций, а также адаптируемости для извлечения комплексных правил позволяет достичь применение методов извлечения временных свойств, основанных на поиске стандартных правил без использования шаблонов [4, 5]. Однако используемые ими модели правил заметно отличаются от выбранной в качестве оптимальной — в виде множества предикатов и в виде модифицированного графа потока управления соответственно.

Полученные результаты позволяют сделать вывод о необходимости дальнейшего поиска автоматизированных подходов к восстановлению правил применения компонентов программных библиотек среди подходов, основанных на методах динамического анализа, которые бы позволили получать более точные и более приближенные к целевой модели спецификации.

Динамические методы извлечения спецификаций

Альтернативой применению рассмотренных в статье [1] подходов к извлечению спецификаций библиотек с использованием методов статического анализа является применение методов, основанных на анализе трасс исполнения программ. Такое решение используется в целом ряде работ [6–17]. Методы динамического анализа в заметной степени отличаются друг от друга и могут быть классифицированы по способу конструирования правил:

1) методы, генерирующие правила применения объектов библиотечного класса [8, 10] (CONTRACTOR++, TEMI), [15, 16];

2) методы, распознающие правила использования таких объектов путем анализа трасс существующих программ [6, 7, 9, 10] (SEKT), [11–14, 17].

С использованием подходов, относящихся к *первой* группе, возможные состояния объектов класса могут быть исследованы путем генерации (по шаблонам) и проверки предикатов. При этом описываться с помощью таких предикатов могут как предусловия, постусловия и инварианты вызовов методов [15, 16], так и абстрактные состояния объектов класса [8, 10] (CONTRACTOR++, TEMI). В качестве входа могут быть использованы как исключительно декларативные описания [8, 10] (CONTRACTOR++), [15], так и трассы вызовов методов исследуемого интерфейса в контексте использующих программ [16], а также их комбинация [10] (TEMI). Большая часть генеративных методов позволяет получать правила, описывающие временные свойства [8] или ограничения на значения состояний целевого объекта и параметры вызовов [15, 16] раздельно. В работе [10] были предложены подходы CONTRACTOR++ и TEMI для построения комплексных правил на основе извлеченной информации. Восстанавливаемые таким образом правила будут представлять собой КА, в которых состояниями являются допустимые комбинации извлеченных предикатов, описывающих условия вызова методов класса, а дугами — разрешенные по предусловиям/постусловиям переходы между ними.

Необходимо отметить, что, в отличие от временных свойств, извлекаемых с применением методов статического анализа и большинства методов, использующих трассы существующих программ [6, 7, 9, 12, 14, 17], выявляемые с использованием методов группы правила будут описывать не стандартные (частые), а разрешающие правила (за исключением метода [16]). Существенным ограничением с точки зрения выразительности извлекаемых спецификаций является то, что получаемые правила могут описывать требования к данным и временным свойствам для объектов единственного выбранного класса. Описание правил взаимодействия объектов разных классов с использованием рассматриваемых решений получено быть не может. Для снижения ресурсоемкости процесса генерации правил используется взаимозависимость предикатов [15, 16], а также вводятся ограничения на длину цепочек [15, 16] или на количество применяемых шаблонов и проверяемых переменных [16]. Для повышения удобства дальнейшего использования полученных результатов дополнительно может быть также осуществлено удаление эквивалентных или более слабых, «поглощаемых» предикатов [15, 16] или удаление предикатов, описывающих несвязанные переменные [16]. В случае с временными свойствами [10] (TEMI) информативность получаемых спецификаций также может быть повышена путем перевода действительно наблюдаемых в трассах переходов из состояния «вероятных» в состояние «актуальных» (при этом соответствующие им целевые состояния дублируются).

Стратегии для извлечения спецификаций с использованием методов *второй* группы заметно варьируются. Их можно разделить на три подгруппы:

1) распознавание КА [6, 7, 13, 17];

2) восстановление автомата с использованием алгоритма k-tails и его модификаций [9, 10] (SEKT), [11];

3) использование методов статистического анализа [12, 14].

В зависимости от реализуемого решения анализируемые трассы могут содержать события

(информацию об объектах, вызываемых методах, параметрах, используемых переменных и соответствии вызовов участкам исходного кода), описывающие последовательность вызовов как в контексте всей программы, использующей библиотеку [6, 10] (SEKT), [13, 14, 17], так и в рамках отдельных ее методов [11, 12], а также события, относящиеся к использованию конкретных объектов [7, 9].

Извлечение спецификаций, осуществляемое на основе методов первой подгруппы [6, 7, 13, 17], предполагает использование событий входной трассы в качестве цепочки символов, которая подается на вход КА, описывающему распознаваемое регулярное выражение. Для каждой последовательности событий, которая потенциально может быть описана с использованием данного выражения, осуществляется проверка того, что прием соответствующей строки переводит автомат в корректное состояние. Результаты распознавания запоминаются, и правило для конкретной последовательности событий считается успешно распознанным при преодолении задаваемого порогового значения количества «принятых» фрагментов трассы. Авторами работы [6] была предложена функция, позволяющая вычислять соответствующие пороговые величины автоматически на основе информации о процессе выявления правил из трассы.

Необходимость в поиске правил, соответствующих автоматам, задаваемым фиксируемыми шаблонами, возникает в связи с тем, что точное распознавание произвольного КА относится к задаче изучения языка и входит в класс NP-полных задач [13]. В рассматриваемых работах для поиска правил применяются шаблонные автоматы с длиной алфавита 2–3 символа. Авторами метода [17] были предложены правила формирования автоматов с произвольным количеством состояний и символов распознаваемого алфавита за счет последовательного объединения «микропаттернов», выявленных с применением таких шаблонов. Для дополнительного снижения ресурсоемкости могут быть использованы оптимизация процесса распознавания за счет бинарных решающих диаграмм (BDD) и методов символьного анализа [13, 17], а также ограничение множества проверяемых автоматов рамками фиксированного окна событий [6].

С помощью методов данной подгруппы могут быть извлечены только временные свойства, описывающие в виде КА стандартные последовательности вызовов методов, принадлежащих интерфейсу изучаемой библиотеки. При этом необходимо отметить, что в статье [11] было показано, что точное восстановление комплексных спецификаций с применением методов композиции к раздельно выявленным с использованием конечно-автоматной модели временным свойствам и ограничениям на область допустимых значений переменных функции в общем случае является невозможным. Ограничения на принадлежность методов, вызовы которых описываются с использованием извлекаемых правил, объектам единственного класса отсутствуют.

Все методы [9, 10] (SEKT), [11], составляющие вторую подгруппу, основаны на использовании алгоритма k-tails [9] или его модификации gk-tail [10] (SEKT), [11]. С их применением могут быть получены как временные свойства [9], так и комплексные правила [10] (SEKT), [11], описывающие допустимые варианты использования объектов библиотечных классов в виде КА с произвольным числом состояний и нефиксированными вариантами переходов. Алгоритм gk-tail включает в себя реализацию базового алгоритма k-tail, дополняя ее выводом и анализом ограничений на данные, и может быть представлен в виде следующей последовательности шагов:

1) поиск последовательностей вызовов, отличающихся только значениями связанных с ними данных;

2) извлечение предикатов с применением метода [16], генерирующего правила применения объектов библиотечного класса;

3) формирование искусственного общего начального состояния восстанавливаемого автомата;

4) поиск и объединение состояний, имеющих эквивалентные цепочки из k исходящих переходов (k-future). При этом k является произвольно задаваемой константой, а требования к эквивалентности определяются как необходимость вызова одних и тех же методов или включение цепочки из k вызовов методов одной последовательности в цепочку из k вызовов методов другой, а также наличие взаимного соответствия между предикатами, описывающими эти вызовы (вызовы равной или включающей цепочки также могут быть описаны с использованием более обобщенных предикатов).

К достоинствам методов рассматриваемой подгруппы относятся возможность вывода «истинных» комплексных правил [10] (SEKT), [11], отсутствие необходимости использования шаблонов при распознавании временных свойств, включаемых в их состав, а также отсутствие принципиального ограничения на принадлежность методов объектам разных классов. Наиболее «узким местом» для методов данной группы будет чувствительность к качеству исходных трасс: любая последовательность событий, подаваемая на вход алгоритму, должна быть генерируема с применением воссоздаваемого автомата. В связи с этим применение алгоритма будет давать некорректные результаты в случае работы

с трассами, содержащими случайное множество событий, не принадлежащих одному правилу, а также с зашумленными трассами. В целях уменьшения влияния рассматриваемых факторов могут быть использованы фрагментирование трассы с вычленением сценария использования метода как упорядоченной последовательности вызовов методов, связанных с искомым по данным, и анализ частоты использования того или иного перехода с последующим устранением редко применяемых переходов.

Последнюю подгруппу методов динамического анализа, распознающих правила использования объектов библиотечного класса путем анализа трасс существующих программ, составляют методы, предполагающие использование техник статистического анализа [12, 14]. Ключевым отличием от методов первой подгруппы, в которых применяется «наивный подсчет», является отсутствие распознавания с хранением состояний шаблонного автомата. Методы данной подгруппы позволяют извлекать временные свойства в виде КА [12] или частых последовательностей [14] вызовов методов библиотечных классов произвольной длины. Извлекаемые правила при этом будут описывать стандартные сценарии использования методов библиотеки.

Для поиска правил применяются как более простые статистические методы [12], основанные на подсчете частоты появления экземпляров того или иного паттерна и проверке преодоления соответствующего порогового значения, так и более сложные [14], использующие Apriori-подобные алгоритмы. Реализация первого [12] из указанных подходов предлагает рассматривать паттерн как набор абстрактных «объектов-ролей» и соответствующих им неупорядоченных множеств вызовов методов библиотечных классов в рамках единственного метода клиентской программы. Выбранный вариант определения накладывает жесткие ограничения на точность и выразительность извлекаемых правил. Так, клиентские методы, реализация которых использует более чем одно правило, будут либо отвергнуты при подсчете, либо использованы для извлечения сложных правил (с потерей возможности восстановления потенциально более полезных простых правил, входящих в их состав). Для повышения качества результатов, получаемых с использованием рассматриваемого решения, применяются приведение задействованных объектов к суперклассам (обладающим требуемым интерфейсом), создание единственного искусственного объекта при анализе работы с коллекциями, а также механизм фильтрации рассматриваемых вызовов по принадлежности к конкретным модулям. Выбор же Apriori-подобного алгоритма [14] для поиска правил изначально позволяет обеспечить определенную устойчивость к шуму во входных трассах, в связи с чем для него подобная предобработка не требуется.

Подводя итог, необходимо отметить, что ввиду причин, аналогичных рассмотренным для подходов, основанных на применении методов статического анализа, проведение сравнения производительности подходов, использующих методы динамического анализа, и качества извлекаемых с их помощью спецификаций является непростой задачей. Так, ряд работ [6, 8, 9, 11, 13, 14] в принципе не содержит информации о точности извлечения, часть работ [7, 15] содержит информацию о качестве правил, полученных в результате отдельных случаев применения средств, реализованных с использованием соответствующих методов, а некоторые авторы вводят свои определения для понятия истинности выведенной спецификации [12, 17].

В целом совокупности рассмотренных методов динамического анализа присущи следующие свойства:

— сравнительно небольшое количество извлекаемых правил;

— разные показатели качества извлекаемых правил;

— использование исходного кода для увеличения точности получаемых результатов.

Среднее количество правил, получаемых с использованием методов динамического анализа, составляет несколько десятков (для сравнения, в случае применения методов статического анализа соответствующее значение может достигать тысяч и миллионов [1]). Это обусловливается использованием рассмотренных механизмов предварительной обработки трасс, а также различных способов фильтрации и композиции найденных правил [7, 17].

Информация о показателях точности результатов, получаемых с использованием методов динамического анализа, представлена в табл. 1–3.

Как видно из табл. 1, для извлечения декларативных спецификаций применяются генеративные методы динамического анализа. Их использование позволяет получать ограничения на значения состояний целевого объекта и параметров вызываемых методов с высокой точностью 92–95 % (для сравнения, точность результирующих спецификаций данного вида, получаемых с применением единственного метода, основанного на статическом анализе, составляет менее 70 % [19]). При этом необходимо отметить, что для подхода [15] оценка показателей качества приведена для результатов анализа весьма незначительного объема кода (10 классов). К недостаткам рассматриваемой группы методов, за исключением общей ограниченности распознаваемых правил шаблонами, можно отнести необходимость их

■ ***Таблица 1***. Подходы для извлечения ограничений на значения состояний целевого объекта и параметры вызываемых методов

■ ***Table 1***. Approaches to mining of rules containing jointly containing predicates for allowed target object's states and method parameters

Работа	Элементы моделей	Точность, %
[15]	Аксиомы, описывающие допустимые диапазоны значений для параметров и результатов вызова методов и последовательностей методов объекта библиотечного класса, фиксируемые с использованием шаблонов	92
[16]	Предусловия и постусловия для методов и инвариантов для библиотечного класса, фиксируемые с использованием шаблонов	95

■ ***Таблица 2***. Подходы для извлечения временных свойств

■ ***Table 2***. Approaches to mining of rules containing temporal properties

Работа	Вид временных свойств	Элементы моделей	Вид статистических методов
[6]	КА с фиксацией стандартных последовательностей вызовов	2-буквенные КА, описывающие стандартные последовательности вызовов на объектах библиотечных классов, фиксируемые с использованием шаблонов	Пороговые методы (частота пар, частота пар с учетом статического контекста вызовов)
[7]			Пороговые методы (частота пар и аномалий)
[8]	КА с фиксацией разрешенных и запрещенных последовательностей вызовов методов	Последовательности допустимых (не приводящих к генерации исключения) вызовов методов на объекте библиотечного класса	Не используются
[9]	Недетерминированный КА с фиксацией стандартных последовательностей вызовов	КА, описывающие стандартные сценарии использования для задаваемых пользователем искомых методов	Используются в постобработке для удаления редко используемых переходов распознанного КА
[12]	Аналогично [6] с указанием частоты вызовов	Аналогично [9]; нельзя зафиксировать разные цепочки, приводящие к вызову одного метода	Пороговые методы (статическая и динамическая частота паттернов)
[13]	Аналогично [6]	Аналогично [6], дополнительно поддерживаются 3-буквенные КА	Пороговые (динамическая частота, с допущением на шум)
[14]	Частые упорядоченные множества вызовов методов библиотечных классов	Стандартные последовательности вызовов методов на объектах классов произвольной длины, предваряющие вызов фиксированного метода	Семейство методов Apriori
[17]	Аналогично [6]	КА, получаемый путем композиции «микропаттернов» [13]	Аналогично [13]

точной настройки (выбор подмножества), а также ручного ввода описания взаимозависимостей с иными шаблонами для уменьшения количества извлекаемых спецификаций (в случае использования нестандартного шаблона; таков алгоритм работы, например, с широко используемым в индустрии инструментальным средством Daikon, описанным в работе [16]).

Аналогично рассмотренному ранее для подходов, основанных на применении методов статического анализа, наиболее многочисленное множество составляют подходы, позволяющие извлекать временные свойства (табл. 2).

Ввиду отсутствия результатов систематической оценки качества получаемых спецификаций, произвести сравнительную оценку соответствующих методов возможно лишь качественно. За исключением метода [8], все они позволяют извлекать стандартные, а не допустимые последовательности вызовов, и относятся к группе методов,

■ ***Таблица 3.*** Подходы для извлечения комплексных свойств
■ ***Table 3.*** Approaches to mining of complex rules

Работа	Вид временных свойств	Элементы моделей	Точность, %
[10] (CONTRACTOR++)	КА с фиксацией разрешенных последовательностей вызовов методов на объекте библиотечного класса	Произвольные потенциально допустимые последовательности вызовов методов на объекте библиотечного класса и ограничения на данные	97
[10] (SEKT)	КА с фиксацией стандартных последовательностей вызовов методов на объектах библиотечных классов, также фиксируются ограничения на параметры методов	Стандартные последовательности вызовов методов на объектах библиотечных классов и ограничения на данные	100
[10] (TEMI)	Аналогично [10] (CONTRACTOR++), два вида переходов — потенциально возможные и подтвержденные	Аналогично [10] (CONTRACTOR++)	99
[11]	Аналогично [10] (SEKT)	Аналогично [10] (SEKT)	–

распознающих правила использования объектов путем анализа трасс программ. При этом принципиальное ограничение на принадлежность вызовов, включаемых в состав правил, объекту конкретного класса отсутствует. Использование метода [8], дающего несколько большую гибкость с точки зрения вида формируемых правил, предполагает необходимость выполнения пользователем ручной работы на нескольких этапах формирования спецификации: выбор предикатов для абстракции внутреннего состояния класса и исключения, относительно которого изучается интерфейс, осуществляется вручную.

Заметная часть подходов [10] (CONTRACTOR++, TEMI), позволяющих извлекать комплексные свойства, которые представлены в табл. 3, также основана на непосредственном применении генеративных методов при исследовании пространства состояний класса, при этом средняя точность результатов, получаемых с их использованием, составляет 97–99 %. Правила аналогичного качества позволяет получать и основанный на восстановлении КА алгоритм [10] (SEKT) (для работы [11] информация о точности сформированных правил не приводится).

Во многом такие значения могут быть объяснены тем, что все обозреваемые подходы используют в процессе своей работы рассмотренное ранее инструментальное средство [16]. Однако, наследуя высокие значения показателей качества получаемых результатов, они наследуют и налагаемые на них ограничения: ограниченность выразительности спецификаций подмножеством шаблонов и необходимость аккуратной ручной работы по осуществлению конфигурирования. Помимо этого, правила, извлекаемые с использованием методов [10] (CONTRACTOR++, TEMI), [11], ограничены описанием временных свойств для объектов, принадлежащих конкретному классу изучаемого библиотечного API. Для подхода [10] (SEKT) такое ограничение выполняться не будет.

Другим способом увеличения качества результирующих спецификаций является осуществление анализа исходного кода для уточнения контекста поиска правил. Так, в методах динамического анализа [6, 12] частота использования паттернов оценивается с учетом статической информации о количестве уникальных мест вызовов. Также с рассматриваемой целью может быть осуществлена предварительная фильтрация трасс, например, для исключения из рассмотрения правил, содержащих вызовы методов-оберток и вложенных методов [7], или выделения методов, принадлежащих одному пакету [12]. Необходимо отметить, что большинство рассматриваемых методов требует наличия распознаваемой трассы и на момент написания статьи используется только при реализации методов второй группы. Рассмотренные ранее для подходов, относящихся к статическому анализу, алгоритмы предварительной фильтрации, основанные на знании о языке или компонентах (удаление вызовов стандартных функций или цепочек get-методов, используемых для получения значений конкретных полей [20] и др.), на момент написания статьи ни одним из обозреваемых методов не реализуются. Предварительная обработка трасс с использованием метрического покрытия, вычисляемого на основе артефактов процесса разработки [18], применение которой позволяло получить правила с наилучшей точностью, также не поддерживается.

Заключение

Извлечение спецификаций является сравнительно новой областью, и большая часть изученных подходов носит экспериментальный или исследовательский характер, поэтому, помимо показателей качества получаемых результатов (систематические данные о которых, зачастую, отсутствуют), в качестве равных по степени важности критериев рассматривались выразительные возможности модели, поддерживаемой подходом (в границах, заданных формализмом [2]), степень участия пользователя в формировании входных и проверке выходных данных и потенциальная сложность адаптации для поддержки нового вида правил (в связи с этим для наиболее пристального изучения были выбраны комплексные и временные свойства).

Наибольшая часть имеющихся на данный момент решений посвящена формированию временных свойств (поведенческих спецификаций) в виде последовательностей вызовов с использованием как методов статического (рассмотренных ранее в работе [1]), так и динамического [6–9, 12–14, 17] анализа. Рассмотренные методы позволяют извлекать правила как с применением шаблонов, так и без, однако с их использованием можно захватывать только стандартные (частые) последовательности вызовов (за исключением метода [8]). Точность результатов, получаемых с использованием методов динамического анализа, будет варьироваться от 18 до 60 %, однако необходимо отметить, что в заметном количестве работ для оценки используются оригинальные критерии, разработанные авторами исследований, что делает их сравнение малоинформативным.

Существенно меньшее количество подходов позволяет захватывать правила с использованием наиболее выразительных комплексных моделей [10] (SEKT, CONTRACTOR++, TEMI), [11]. Как статические, так и динамические подходы позволяют захватывать комплексные правила в свободной форме. Для большинства методов, базирующихся на динамическом анализе [10] (SEKT, CONTRACTOR++, TEMI), значение точности результирующих спецификаций будет составлять 97–100 %. Наиболее точные разрешающие комплексные спецификации (обладающие качеством более 90 %) могут быть получены с использованием генеративных методов динамического анализа. При этом следует подчеркнуть, что методы подходов, позволяющих достичь таких результатов, генерируют правила, используя в качестве входа предикаты, описывающие ограничения на данные, которые также были извлечены с использованием генеративных методов. К ограничениям рассматриваемых подходов относятся невозможность восстановления с их использованием спецификаций, описывающих взаимодействие нескольких объектов (что позволяет сделать подход [3], основанный на использовании методов статического анализа), изменение состояния библиотеки и семантику вызовов, а также использование шаблонов.

Таким образом, по результатам проведенного обзора, с точки зрения выразительности и качества получаемых правил, а также степени вовлеченности пользователя, наиболее целесообразным в контексте разработки подхода извлечения поведенческих правил использования элементов библиотечного API с применением комплексной модели, предложенной в работе [2], является разработка генеративных методов динамического анализа, основанных на внешних инструментах (например, [16]) для извлечения ограничений на данные.

Работа частично поддержана стипендиальной программой компании «Сименс» в СПбПУ.

Литература

1. **Егорова И. С., Ицыксон В. М.** Обзор статических методов восстановления частичных спецификаций программных библиотек на основе анализа программных проектов// Информационно-управляющие системы. 2017. № 6. С. 66–75. doi:10.15217/issn1684-8853.2017.6.66
2. **Ицыксон В. М.** Формализм и языковые инструменты для описания семантики программных библиотек // Моделирование и анализ информационных систем. 2016. № 23. С. 754–766.
3. **Ramanathan M., Grama A., Jaganathan S.** Static Inference with Predicate Mining // Proc. of the 28th ACM SIGPLAN Conf. on Programming Language Design and Implementation. 2007. P. 123–134.
4. **Wasylkowski A., Zeller A.** Mining Temporal Specifications from Object Usage // Proc. of the 2009 IEEE/ACM Intern. Conf. on Automated Software Engineering. 2009. P. 295–306.
5. **Nguyen T. T., Nguyen H. A., Pham N. H., Al-Kofahi J. M., Nguyen T. N.** Graph-based Mining of Multiple Object Usage Patterns // Proc. of the 7th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering. 2009. P. 383–392.
6. **Gabel M., Su Z.** Online Inference and Enforcement of Temporal Properties // Proc. of the 32nd ACM/IEEE Intern. Conf. on Software Engineering. 2010. Vol. 1. P. 15–24.
7. **Yang J., Evans D., Bhardwaj D., Bhat T., Das M.** Perracotta: Mining Temporal API Rules from Imperfect

Traces // Proc. of the 28th Intern. Conf. on Software Engineering. 2006. P. 282–291.
8. **Alur R., Cerny P., Madhusudan P., Nam W.** Synthesis of Interface Specifications for Java Classes // Proc. of the 32nd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. 2005. P. 98–109.
9. **Ammons G., Bodik R., Larus J. R.** Mining Specifications // Proc. of the 29th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. 2002. P. 4–16.
10. **Krka I., Medvidovic N., Brun Y.** Automatic Mining of Specifications from Invocation Traces and Method Invariants // Proc. of the 22nd ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering. 2014. P. 178–189.
11. **Lorenzoli D., Mariani L., Pezze M.** Automatic Generation of Software Behavioral Models // Proc. of the 30th Intern. Conf. on Software Engineering. 2008. P. 501–510.
12. **Pradel M., Gross T. L.** Automatic Generation of Object Usage Specifications from Large Method Traces // Proc. of the 2009 IEEE/ACM Intern. Conf. on Automated Software Engineering. 2009. P. 371–382.
13. **Gabel M., Su Z.** Symbolic Mining of Temporal Specifications // Proc. of the 30th Intern. Conf. on Software Engineering. 2008. P. 51–60.
14. **Lo D., Khoo S., Liu C.** Mining Past-Time Temporal Rules from Execution Traces // Proc. of the 2008 Intern. Workshop on Dynamic Analysis: Held in Conjunction with the ACM SIGSOFT Intern. Symp. on Software Testing and Analysis (ISSTA 2008). 2008. P. 50–56.
15. **Henkel J., Diwan A.** Discovering Algebraic Specifications from Java Classes // Proc. of the 17th European Conf. on Object-Oriented Programming, ECOOP 2003, Darmstadt, Germany, July 21–25, 2003. Lecture Notes in Computer Science, 2003. Vol. 2743. P. 431–456.
16. **Ernst M. D., Perkins J. H., Guo P. J., McCamant S., Pacheco C., Tschantz M. S., Xiao C.** The Daikon System for Dynamic Detection of Likely Invariants // Science of Computer Programming. 2007. Vol. 69. Iss. 1–3. P. 35–45.
17. **Gabel M., Su Z.** Javert: Fully Automatic Mining of General Temporal Properties from Dynamic Traces // Proc. of the 16th ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering. 2008. P. 339–349.
18. **Goues C., Weimer W.** Specification Mining with Few False Positives // Proc. of the 15th Intern. Conf. on Tools and Algorithms for the Construction and Analysis of Systems: Held as Part of the Joint European Conf. on Theory and Practice of Software. 2009. P. 292–306.
19. **Thummalapenta S., Xie T.** Alattin: Mining Alternative Patterns for Defect Detection // Proc. of the 2009 IEEE/ACM Intern. Conf. on Automated Software Engineering. 2009. P. 293–323.
20. **Livshits B., Zimmerman T.** DynaMine: Finding Common Error Patterns by Mining Software Revision Histories // Proc. of the 10th European Software Engineering Conf. Held Jointly with 13th ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering. 2005. P. 296–305.

UDC 004.4′2
doi:10.15217/issn1684-8853.2018.2.67

Review of Dynamic Methods for Extraction of Partial Software Library Specification

Egorova I. S.[a], Post-Graduate Student, is.egorova@mail.ru
Itsykson V. M.[a], PhD, Tech., Associate Professor, vlad@icc.spbstu.ru
[a]Peter the Great Saint-Petersburg Polytechnic University, 29, Polytechnicheskaya St., 195251, Saint-Petersburg, Russian Federation

Introduction: Nowadays, software development is commonly based on the reuse of previously created components united into software libraries and frameworks. Such third-party components often come without full and precise documentation. Specification recovery can be performed by the analysis of successfully developed open-source projects. **Purpose:** Analysis and classification of the most prospective approaches to automated extraction of software library specifications based on the methods of dynamic code analysis. **Results:** Various approaches used to describe library components have been reviewed and compared. We have discussed the ways of deriving specifications using dynamic analysis methods. It is found out that the most exact dynamic analysis approaches are based on algorithms which generate and check template hypotheses. Their main restrictions are the lack of possibility to specify interaction between several objects and changes in the library state, and also the usage of templates. Besides, it should be noted that no one of the reviewed approaches currently supports the recovery of semantic description of library method calls.

Keywords — Software Library Specification, Formal Specification, Dynamic Extraction of Specifications, Extended Finite State Automata, Temporal Properties, Behavioral Software Library Model, Complex Rules, Dynamic Analysis.

Citation: Egorova I. S., Itsykson V. M. Review of Dynamic Methods for Extraction of Partial Software Library Specification. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2018, no. 2, pp. 67–75 (In Russian). doi:10.15217/issn1684-8853.2018.2.67

References

1. Egorova I. S., Itsykson V. M. Survey of Static Methods for Partial Software Library Specification Extraction. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2017, no. 6, pp. 66–75 (In Russian). doi:10.15217/issn1684-8853.2017.6.66
2. Itsykson V. M. The Formalism and Language Tools for Semantics Specification of Software Libraries. *Modelirovanie i analiz informatsionnykh sistem* [Modeling and Analysis of Information Systems], 2016, no. 23, pp. 754–766 (In Russian).
3. Ramanathan M., Grama A., Jaganathan S. Static Inference with Predicate Mining. *Proc. of the 28th ACM SIGPLAN Conf. on Programming Language Design and Implementation*, 2007, pp. 123–134.
4. Wasylkowski A., Zeller A. Mining Temporal Specifications from Object Usage. *Proc. of the 2009 IEEE/ACM Intern. Conf. on Automated Software Engineering*, 2009, pp. 295–306.
5. Nguyen T. T., Nguyen H. A., Pham N. H., Al-Kofahi J. M., Nguyen T. N. Graph-based Mining of Multiple Object Usage Patterns. *Proc. of the 7th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering*, 2009, pp. 383–392.
6. Gabel M., Su Z. Online Inference and Enforcement of Temporal Properties. *Proc. of the 32nd ACM/IEEE Intern. Conf. on Software Engineering*, 2010, vol. 1, pp. 15–24.
7. Yang J., Evans D., Bhardwaj D., Bhat T., Das M. Perracotta: Mining Temporal API Rules from Imperfect Traces. *Proc. of the 28th Intern. Conf. on Software Engineering*, 2006, pp. 282–291.
8. Alur R., Cerny P., Madhusudan P., Nam W. Synthesis of Interface Specifications for Java Classes. *Proc. of the 32nd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages*, 2005, pp. 98–109.
9. Ammons G., Bodik R., Larus J. R. Mining Specifications. *Proc. of the 29th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages*, 2002, pp. 4–16.
10. Krka I., Medvidovic N., Brun Y. Automatic Mining of Specifications from Invocation Traces and Method Invariants. *Proc. of the 22nd ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering*, 2014, pp. 178–189.
11. Lorenzoli D., Mariani L., Pezze M. Automatic Generation of Software Behavioral Models. *Proc. of the 30th Intern. Conf. on Software Engineering*, 2008, pp. 501–510.
12. Pradel M., Gross T. L. Automatic Generation of Object Usage Specifications from Large Method Traces. *Proc. of the 2009 IEEE/ACM Intern. Conf. on Automated Software Engineering*, 2009, pp. 371–382.
13. Gabel M., Su Z. Symbolic Mining of Temporal Specifications. *Proc. of the 30th Intern. Conf. on Software Engineering*, 2008, pp. 51–60.
14. Lo D., Khoo S., Liu C. Mining Past-Time Temporal Rules from Execution Traces. *Proc. of the 2008 Intern. Workshop on Dynamic Analysis: Held in Conjunction with the ACM SIGSOFT Intern. Symp. on Software Testing and Analysis (ISSTA 2008)*, 2008, pp. 50–56.
15. Henkel J., Diwan A. Discovering Algebraic Specifications from Java Classes. *Proc. of the 17th European Conf. on Object-Oriented Programming, ECOOP 2003*, Darmstadt, Germany, July 21–25, 2003, Lecture Notes in Computer Science, 2003, vol. 2743, pp. 431–456.
16. Ernst M. D., Perkins J. H., Guo P. J., McCamant S., Pacheco C., Tschantz M. S., Xiao C. The Daikon System for Dynamic Detection of Likely Invariants. *Science of Computer Programming*, 2007, vol. 69, iss. 1–3, pp. 35–45.
17. Gabel M., Su Z. Javert: Fully Automatic Mining of General Temporal Properties from Dynamic Traces. *Proc. of the 16th ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering*, 2008, pp. 339–349.
18. Goues C., Weimer W. Specification Mining with Few False Positives. *Proc. of the 15th Intern. Conf. on Tools and Algorithms for the Construction and Analysis of Systems: Held as Part of the Joint European Conf. on Theory and Practice of Software*, 2009, pp. 292–306.
19. Thummalapenta S., Xie T. Alattin: Mining Alternative Patterns for Defect Detection. *Proc. of the 2009 IEEE/ACM Intern. Conf. on Automated Software Engineering*, 2009, pp. 293–323.
20. Livshits B., Zimmerman T. DynaMine: Finding Common Error Patterns by Mining Software Revision Histories. *Proc. of the 10th European Software Engineering Conf. Held Jointly with 13th ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering*, 2005, pp. 296–305.