Peshehonov Vladimir Grigorjevich – JSC Concern Central Scientific and Research Institute "Elektropribor"; e-mail: office@eprib.ru; 30, Malaya Posadskaya street, Saint Petersburg, 197046, Russia; phone: +78122323376; **g**eneral director; dr. of eng. sc.; professor.

Braga Yurii Alekseevich – e-mail: ybraga@mail.ru; тел.: 89312039867; the deputy chief of the research department; cand. of eng. sc; associate professor.

Mashoshin Andrey Ivanovich – e-mail: amashoshin@eprib.ru; phone: +79217632345; chief of the research department; dr. of eng. sc.; professor.

УДК 004.415.53

П.П. Кравченко, С.В. Бирюков

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ПРОГРАММНЫХ ИНТЕРФЕЙСОВ ПРИЛОЖЕНИЙ: МЕТОД И СРЕДСТВА

Рассматривается задача сокращения трудоемкости тестирования программных интерфейсов. Предлагается разработанный авторами метод автоматизации тестирования программных интерфейсов, основанный на автоматическом разборе спецификации интерфейса, построении унифицированной модели и обходе модели для генерации набора тестов. Приведена структура программной системы APITest, в которой реализован разработанный метод. Рассматривается применение метода и программной системы для решения практических задач тестирования интерфейсов.

Программный интерфейс приложения; спецификация интерфейса; унифицированная модель; тестирование программного обеспечения.

P.P. Kravchenko, S.V. Biryukov

AUTOMATED TESTING OF APPLICATION PROGRAMMING INTERFACES: METHOD AND SOFTWARE

The paper considers the problem of reducing the complexity of testing software interfaces. It is proposed by the authors developed a method of test automation software interfaces based on automatic analysis of interface specifications, the construction of a unified model and the bypass model to generate a set of tests. Describes the structure of the software system APITest, which implements the method. The application of the method and software system for solving practical problems of testing interfaces.

Application programming interface; interface specification; unified model; software testing.

Введение. В настоящее время необходимость систематизированного тестирования в промышленной разработке программного обеспечения (ПО) общепризнанна и неоспорима. Традиционные методы разработки тестов вручную уже не могут обеспечить быстрое и качественное тестирование современных программных систем. Однако в большинстве случаев тестируемое ПО связывают с наличием в нем графического интерфейса, позволяющего задавать входные воздействия на целевую систему и получать результаты. Зачастую при этом остается без внимания целый класс ПО, предоставляющий доступ к своей функциональности лишь посредством программного интерфейса.

Под программным интерфейсом приложения или интерфейсом программирования (Application Programming Interface, API) понимается набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. Детали реализации API при этом, как правило, скрыты. Примерами API могут служить библиотеки функций (COM DLL, .NET assembly), web-сервисы, встроенные средства программирования приложений (VBA в MS Office).

Среди существующих подходов к автоматизации тестирования программных интерфейсов выделяются метод разработки специализированных программ тестирования – тестовых драйверов и утилит [1] – и тестирование на основе модели ПО [2, 3]. Использование специализированных программ тестирования затруднительно для приложений с большим количеством интерфейсных функций, поскольку при изменениях в интерфейсе возрастает трудоемкость модификации программы тестирования. Кроме того, код программы тестирования может быть достаточно сложным, возникает проблема необходимости верификации самой программы тестирования.

Существующие подходы к автоматизации тестирования на основе специализированной модели ПО требуют ручного описания модели, разработки процедур преобразования модельных операций в реальные тестовые вызовы, создания итераторов входных воздействий. Указанные операции являются трудоемкими, что увеличивает время тестирования.

Таким образом, разработка метода автоматизации тестирования программного интерфейса приложения, обеспечивающего на основе унифицированной модели интерфейса сокращение трудоемкости тестирования, является актуальной проблемой.

Метод и алгоритмы автоматизации тестирования программных интерфейсов. Задача автоматизации тестирования API является комплексной, включающей в себя автоматизацию генерации, выполнения тестов, анализа результатов работы интерфейсных методов. Автоматизация тестирования преследует, по крайней мере, две цели, такие как сокращение трудоемкости и повышение качества тестирования. Среди особенностей тестирования интерфейсов программирования приложений можно выделить отсутствие оболочки для осуществления входных тестовых воздействий и получения результатов, зависимость результата работы функции от комбинации параметров, необходимость использования параметров сложного типа, построение цепочек вызова функций для получения определенного результата.

Одним из основных недостатков существующих методов автоматизации тестирования на основе моделей является необходимость описания модели вручную. При этом затраты ресурсов на построение модели могут нивелировать преимущества автоматизации. Для сокращения трудоемкости тестирования предложено автоматизировать этап построения модели, используя полученную при проектировании программной системы спецификацию API. Рассмотрены различные форматы представления спецификаций интерфейсов: язык описания интерфейсом IDL, диаграмма классов UML и метаданные библиотеки .NET. Форматы имеют примерно одинаковый уровень абстракции, что позволяет использовать любой из них для построения модели интерфейса программирования.

Указанная идея использована при разработке метода автоматизации тестирования программных интерфейсов. Совокупность действий метода представлена в виде UML-диаграммы последовательности (рис 1).

Модуль построения модели API (построитель) производит автоматический разбор спецификации и построение унифицированной модели интерфейса. Модуль обхода модели (обходчик) выполняет удаление циклов в графе модели и добавляет определенные по умолчанию домены значений для объектов интерфейса. Затем тестировщик расширяет модель сущностями, отражающими поведение объектов интерфейса: специальными значениями и конструкторами объектов, формализованными функциональными требованиями. Выполняется ее обход и генерация тестов в виде лексем, удобных для трансляции в последовательности вызовов интерфейсных функций на языке программирования. Полученные тесты могут содержать вызовы, не увеличивающие степень тестового покрытия, поэтому подвер-

гаются оптимизации. Модуль оптимизации тестов (оптимизатор) избавляет набор от избыточности. Далее модуль генерации тестов (генератор тестов) выполняет трансляцию лексем в код вызовов интерфейсных функций, используя знания о специфике языка: синтаксисе, типах данных, константах и т.д. Полученные тесты могут быть выполнены в одном из многочисленных существующих средств автоматизации выполнения тестов, например семейство программных пакетов xUnit.

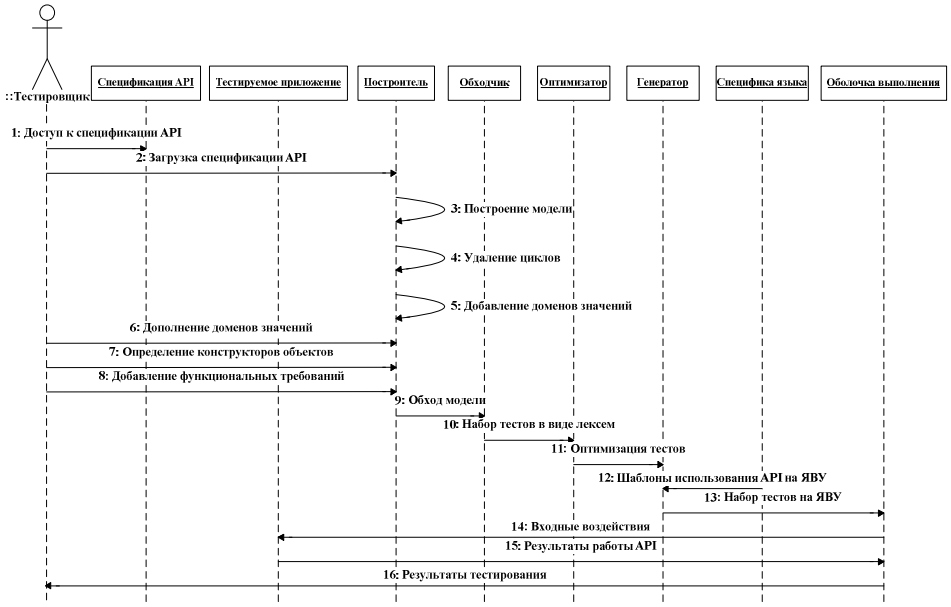

Рис. 1. Диаграмма последовательности метода автоматизации тестирования API

Разработан формат унифицированной модели программного интерфейса, которая может быть построена путем разбора спецификаций в различных нотациях: IDL, UML, NET-сборок. Унифицированная модель интерфейса представляет собой ориентированный граф. В нем выделяются вершины двух типов – объекты интерфейса и действия над ними (интерфейсные методы). Вершины первого типа соответствуют объектам, причем выделяются сложные и простые объекты. Простые объекты – это элементарные типы данных среды использования API (языка программирования). Сложные объекты задекларированы непосредственно в интерфейсе. Их создание, удаление и операции над ними осуществляются посредством интерфейсных методов. Вершины второго типа соответствуют действиям над объектами. Среди них также различается две категории – атрибуты и методы. Атрибуты являются интерфейсом для получения или назначения некоторой характеристики объекта. Некоторые атрибуты доступны только для чтения данных, их модификация скрыта внутри объекта. Методы выполняют манипуляции над объектом в зависимости от набора входных параметров и позволяют получить результат (возможно пустой) – потенциально полезные стороннему пользователю API данные. Пример графа модели представлен на рис. 2.

Дуги графа всегда соединяют вершины разных типов. Таким образом, в графе присутствуют только дуги «объект-действие» и «действие-объект». При этом направление дуги показывает направление потока данных при выполнении действия.

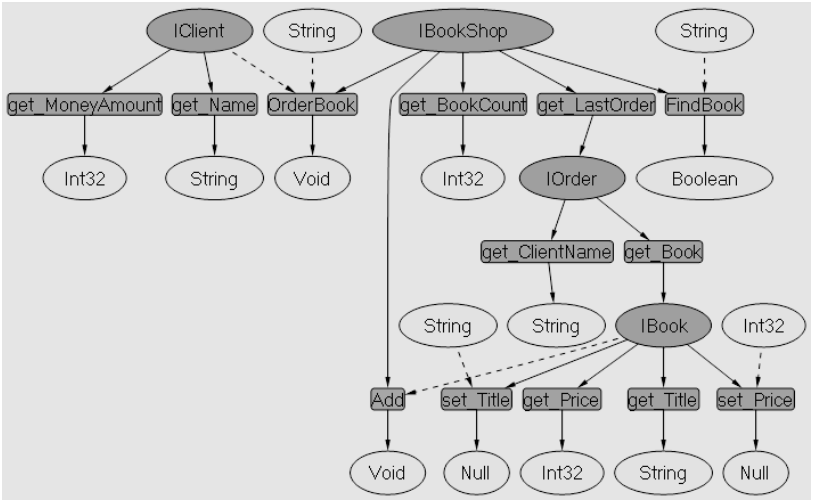

Рис. 2. Пример графа унифицированной модели API

Для реализации предложенного метода был разработан ряд алгоритмов: алгоритм построения унифицированной модели на основе автоматического разбора спецификации; алгоритм обхода унифицированной модели для генерации набора тестов в виде последовательности лексем; алгоритм оптимизации на основе статического анализа набора тестов без изменения степени тестового покрытия модели; алгоритм трансляции трасс тестов в виде лексем в код на языке программирования.

Предложен способ расширения модели функциональными требованиями, состоящий из определения доменов значений простых объектов и формализации требований и ограничений с использованием метода контрактных спецификаций [4]. Для каждого простого объекта модели, не имеющего входящих ребер, должен быть определен набор значений, которые будут подставляться при использовании этого объекта в коде вызова интерфейсных функций. Часть значений может быть заранее определена в шаблоне, а их добавление в домен тестовых значений объекта возможно автоматически.

В контексте метода автоматизации тестирования программных интерфейсов контрактные спецификации (пред- и постусловия) имеют вид утверждений (assert). Утверждение представляет собой логическое выражение. Если утверждение истинно, реализованная функциональность метода является корректной. Предусловие накладывается на действие и проверяется перед его выполнением. Действие будет выполнено, если все его предусловия разрешают выполнение. Постусловие также накладывается на действие, но проверяется после его выполнения. Если постусловие истинно, работа интерфейсного метода соответствует заданному этим условием функциональному требованию. Поскольку функциональные требования хорошо привязаны к реализации тестовых сценариев в виде кода вызовов, они могут быть использованы для автоматической генерации оракулов.

Программная реализация метода автоматизации тестирования программных интерфейсов. Разработанный метод реализован в виде универсальной среды тестирования программных интерфейсов APITest [5]. APITest состоит из независимых программных модулей (подсистем), объединенных единым графическим интерфейсом пользователя. Структурная схема программной реализации представлена на рис. 3.

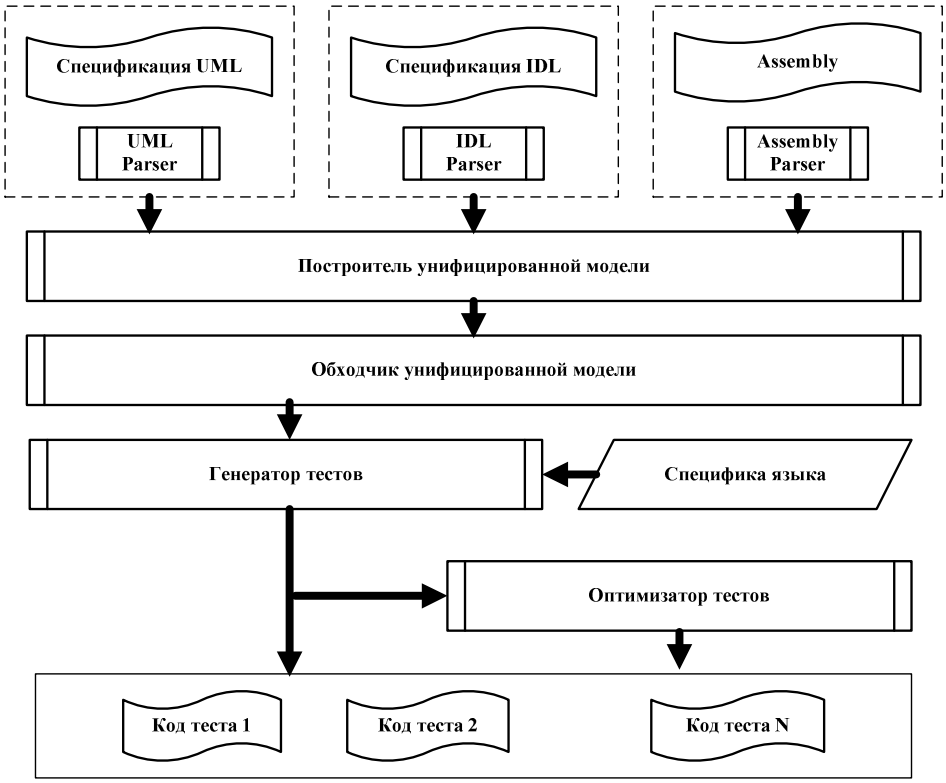

Рис. 3. Структурная схема среды тестирования программных интерфейсов APITest

Программная система APITest реализует последовательность действий метода автоматизации тестирования API, представленную на рис. 1. Стрелками указано направление потока данных при взаимодействии отдельных подсистем.

В качестве входных данных APITest использует спецификацию интерфейса программирования – формализованное описание элементов интерфейса, их параметров, способов связи и взаимодействия с другими объектами. Как было установлено, наиболее распространено описание API, заданное в терминах языка описания интерфейсов IDL (Interface Description Language) или диаграммы классов языка моделирования UML (Unified Model Language). В качестве спецификации может быть также рассмотрен манифест динамической библиотеки. Для каждого из указанных типов спецификаций в программе присутствует подпрограмма-анализатор, производящая автоматический разбор спецификации конкретного типа и передающая данные построителю унифицированной модели API. В систему могут быть интегрированы другие виды спецификаций и анализаторов.

Построитель унифицированной модели по информации от анализатора строит модель программного интерфейса в виде графа, абстрагированную от конкретной спецификации. Функциональность построителя дополнена возможностью вывода графа модели на экран в удобной для пользователя форме.

Программная система имеет графический интерфейс редактирования доменов значений простых объектов, а также определения для объектов и операции предусловий и постусловий. Для удобства и наглядности редактирование происходит с привязкой к модели интерфейса программирования.

Построитель передает указатель на структуру данных, содержащую представление графа модели в памяти, в обходчик унифицированной модели. Основная задача обходчика – генерация набора путей в графе, удовлетворяющих некоторому критерию вычислимости объектов. В систему могут быть интегрированы другие алгоритмы обхода и критерии вычислимости, если существующие слишком трудоемки ввиду большого размера графа.

Набор сгенерированных путей в виде лексем передается генератору тестов, заменяющему вершины в путях на вызовы конкретных интерфейсных методов. Для формирования окончательного синтаксически правильного кода вызовов генератор использует информацию о специфике языка, в среде которого планируется производить тестирование. В специфике языка перечисляются типы данных, константы, процедуры создания объектов и др. Специфика языка хранится в виде древовидной структуры в формате XML, что позволяет расширить ее другими языками программирования.

Оптимизатор тестов представляет собой опциональную подсистему, реализующую метод удаления не увеличивающих покрытие модели и не приносящих пользы для поиска дефектов тестов.

Система APITest была использована в рамках НИР "Аксиома-ЮФУ" для автоматизации функционального тестирования программных библиотек макета корреляционно-экстремальной системы навигации. Внедрение разработанной программной системы автоматизации тестирования позволило сократить трудоемкость тестирования на 65–75 % за счет автоматизации этапов построения модели, разработки и кодирования тестов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Макгрегор Д., Сайкс Д.* Тестирование объектно-ориентированного программного обеспечения. Практическое пособие. – К.: ООО «ТИД «ДС», 2002. – 432 с.
2. *Кулямин В., Петренко А., Косачев А., Бурдонов И.* Подход UniTesK к разработке тестов // Программирование. – 2003. – № 29 (6). – С. 25-43.
3. *Jacky J., Veanes M., Campbell C., Schulte W.* Model-Based Software Testing and Analysis with C#. – Cambridge University Press, 2008. – 349 p.
4. *Meyer B.* Applying «Design by Contract» // IEEE Computer. – 1992. – Vol. 25, №. 10. – P. 40-51.
5. *Бирюков С.В.* Программная реализация метода автоматизации функционального тестирования интерфейса программирования приложения // Инновационные подходы к применению информационных технологий в профессиональной деятельности. Сборник трудов Второй Международной научно-практической Интернет-конференции Белгородского филиала НАЧОУ ВПО СГА. – Белгород: Изд-во ГиК, 2010. – С. 297-301.

Статью рекомендовал к опубликованию д.т.н., профессор Я.Е. Ромм.

Кравченко Павел Павлович – Технологический институт федерального государственного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге; e-mail: kravch@tsure.ru; 347928, г. Таганрог, пер. Некрасовский, 44; тел.: 88634371673; кафедра математического обеспечения и применения ЭВМ; заведующий кафедрой; д.т.н.; профессор.

Бирюков Сергей Вячеславович – Компания «Программные технологии», г. Таганрог, инженер по программным продуктам; e-mail: birser@mail.ru; 347923, г. Таганрог, ул. Инструментальная, 29/2; тел.: 88634315100; инженер по программным продуктам.

Kravchenko Pavel Pavlovich – Taganrog Institute of Technology – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education “Southern Federal University”; e-mail: kravch@tsure.ru; 44, Nekrasovskiy, Taganrog, 347928, Russia; phone: +78634371673; the department of software engineering; head of department; dr. of eng. sc.; professor.

Biryukov Sergey Vyacheslavovich – «Software technologies»; e-mail: birser@mail.ru; 29-2, Instrumentalnaya street, Taganrog, 347923, Russia; phone: +78634315100; product engineer.