6. *Бондарос Ю., Колоколов А., Костюк А.* Исследование речевых сигналов в условиях кабины летательного аппарата // Вестник компьютерных и информационных технологий. – 2008. – № 4. – С. 2-10.
7. *Лобанов Б., Цирульник Л., Железны М., Крноул З., Ронжин А., Карпов А.* Система аудиовизуального синтеза русской речи // Информатика. – Минск: ОИПИ Беларуси. – 2008. – № 4 (20). – С. 67-78.

Карпов Алексей Анатольевич
Учреждение Российской академии наук Санкт-Петербургский институт информатики и автоматизации РАН.
Email: karpov@iias.spb.su.
199178, г. Санкт-Петербург, 14-я линия, д. 39.
Тел.: 88123287081.

Karpov Alexey Anatolievich
Institution of the Russian Academy of Sciences St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences.
Email: karpov@iias.spb.su
14-th Line, 39, Saint-Petersburg, 199178, Russia.
Phone: 88123287081.

УДК 004.415.53

С.В. Бирюков

РАЗРАБОТКА МЕТОДА АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ СИСТЕМ С ИНТЕРФЕЙСОМ ПРОГРАММИРОВАНИЯ

В данной работе рассматриваются вопросы автоматизации тестирования на основе модели систем с интерфейсом программирования. Особое внимание уделяется вопросам автоматической генерации тестовых сценариев и построения тестовых оракулов. Приводится структура данных для хранения и обработки модели интерфейса с расширением для функциональных требований. Предложенный подход реализован в рамках программной среды генерации тестовых сценариев и оракулов APITest.

Интерфейс программирования; автоматизация тестирования; спецификация интерфейса; унифицированная модель.

S.V. Biryukov

THE DEVELOPMENT OF AUTOMATED TESTING METHOD FOR SYSTEMS WITH PROGRAMMING INTERFACE

The issues of model-based automated testing for systems with programming interface are considered in this paper. Particular attention is paid to the automatic generation of test scenarios and building test oracles. The data structure for storage and processing of interface model with the expansion of functional requirements is offer. The proposed approach is implemented within the software environment for generating test scripts and oracles APITest.

Programming interface; automated testing; interface specification; unified model.

При разработке и сопровождении программных систем (ПС) значительная часть усилий тратится на поиск и устранение ошибок. Самым распространённым методом поиска ошибок является тестирование, т.е. процесс выполнения программ с целью обнаружения ошибок [1]. В настоящее время необходимость систематизированного тестирования в промышленной разработке ПС общепризнанна и неоспорима. Однако в большинстве случаев тестируемые ПС связывают с наличием в нем графического интерфейса пользователя, которое служит медиа-

тором, преобразующим тестовые сценарии в реальные входные воздействия на целевую систему, и представляющим результаты в удобной для восприятия форме. Зачастую при этом остается без должного внимания целый класс ПС, предоставляющий доступ к своей функциональности лишь посредством интерфейса программирования.

Под интерфейсом программирования (Application Programming Interface, API) понимается набор методов (функций), предоставляющих доступ к некоторой функциональности системы на уровне кода, скрывая при этом особенности реализации этой функциональности. Примерами API могут служить библиотеки функций (COM DLL, .NET assembly), web-сервисы, встроенные средства программирования приложений (VBA в MS Office).

Тестирование API обладает рядом особенностей, которые необходимо учитывать при автоматизации. Вокруг интерфейсных методов отсутствует графическая или иная оболочка, позволяющая осуществлять тестовые воздействия и получать результаты этих воздействий. Такую оболочку необходимо создавать отдельно, что приводит к дополнительным затратам ресурсов на тестирование.

Данные для обработки поступают в методы интерфейса посредством передачи их через один или несколько входных параметров. В качестве входных параметров могут использоваться как простые типы данных (целые или вещественные числа, строки, логические значения и т.д.), так и сложные, полученные, как правило, в результате работы других методов. Даже для простых параметров число их возможных значений слишком велико, чтобы проверить работу метода для каждого из них за разумное время. Необходимо выбрать небольшой набор перспективных в плане поиска дефектов значений для каждого из параметров, чтобы протестировать максимум функциональности метода. Кроме того, часть функциональности может зависеть не от каждого параметра в отдельности, а от их комбинации.

Часто методы интерфейса оперируют данными сложного типа в качестве параметров. Эти данные могут быть получены в основном как результат работы других методов. Таким образом, не всегда возможно независимое тестирование интерфейсных методов, необходимо уметь строить последовательности вызовов. Причем промежуточный результат должен быть пригодным для успешного вызова последующего метода в последовательности.

Задача автоматизации тестирования API является комплексной, включающей в себя автоматизацию основных этапов:

- Генерация тестовых сценариев. Каждый тестовый сценарий представляет собой последовательность вызовов интерфейсных методов. При этом должны быть определены реальные значения входных параметров.
- Выполнение тестовых сценариев. Использование интерфейса возможно лишь при наличии оболочки, с помощью которой возможно задавать тестовые воздействия и обрабатывать результаты работы.
- Анализ результатов работы методов. Может быть осуществлен путем сравнения с эталоном или использования тестовых оракулов – процедур, автоматически предсказывающих корректный результат работы методов.

Входными данными для задачи тестирования интерфейса является спецификация интерфейса – формализованное описание элементов интерфейса, их параметров, способов связи и взаимодействия с другими объектами. Спецификация отображает только ту часть информации, которая необходима стороннему приложению для использования интерфейса. Она представляет собой баланс между недостатком и избытком знаний об интерфейсе. Обзор показал, что не существует единого формата спецификации интерфейсов программирования. 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 спецификаций можно также использовать манифест динамической библиотеки. На рис. 1 представлен пример спецификации API в терминах UML.

Спецификация является моделью интерфейса программирования. Моделирование программной системы в общем случае – это способ представления структуры и поведения системы. Модели проще, чем реальные системы, они помогают понять и предсказать поведение. В голове разработчика и тестировщика всегда присутствует та или иная «модель» устройства программы, а также «модель» ее желаемого поведения, исходя из которой, в частности, составляются списки проверяемых свойств и создаются соответствующие тестовые сценарии [2].

Важной частью подхода при тестировании на основе моделей являются тестовые оракулы (oracles). В общем случае оракулом является процедура, описывающая способ определения корректности поведения целевой системы. Формальные или неформальные оракулы используются в любом подходе к тестированию, когда представление о корректном поведении системы сравнивается с полученным в ходе выполнения теста. Описание системы в виде формальной модели позволяет также формализовать оракулы.

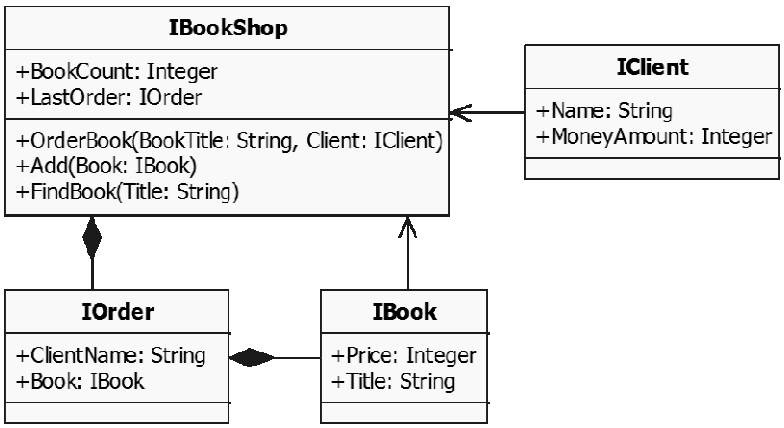

Рис. 1. Пример спецификации API в виде диаграммы классов UML

Заметим, что модель API в одном из представленных выше форматов является структурной (архитектурной). В ней содержится мало информации о поведении системы, необходимо расширение модели знаниями о функциональности. Была разработана универсальная структура данных для представления архитектуры API с расширением для хранения функциональных требований к интерфейсным функциям. Пример структуры для приведенного на рис. 1 интерфейса представлен на рис. 2.

На начальном этапе осуществляется автоматический разбор спецификации конкретного вида и производится построение унифицированной модели. Предлагаемая модель API представляет собой граф с вершинами и ребрами специального вида [3]. В графе различаются 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 типы данных среды использования API (языка программирования). Сложные объекты задекларированы непосредственно в интерфейсе. Их создание и операции с ними осуществляются посредством интерфейсных методов.

Вершины второго типа соответствуют действиям над объектами. Среди них также различаются два типа – свойства и функции. Свойства являются интерфейсом для получения или назначения некоторой характеристики объекта. Некоторые свойства доступны только для чтения данных, их модификация скрыта внутри объекта. Функции выполняют манипуляции над объектом в зависимости от набора входных параметров и позволяют получить результат (возможно пустой) – потенциально полезные стороннему пользователю API-данные.

Ориентированные ребра графа всегда соединяют вершину-объект и вершину-действие. При этом ориентация ребра показывает направление потока данных при выполнении действия. Таким образом, для каждого действия в графе легко доступен объект, выполняющий действие, а также объекты-аргументы функций интерфейса.

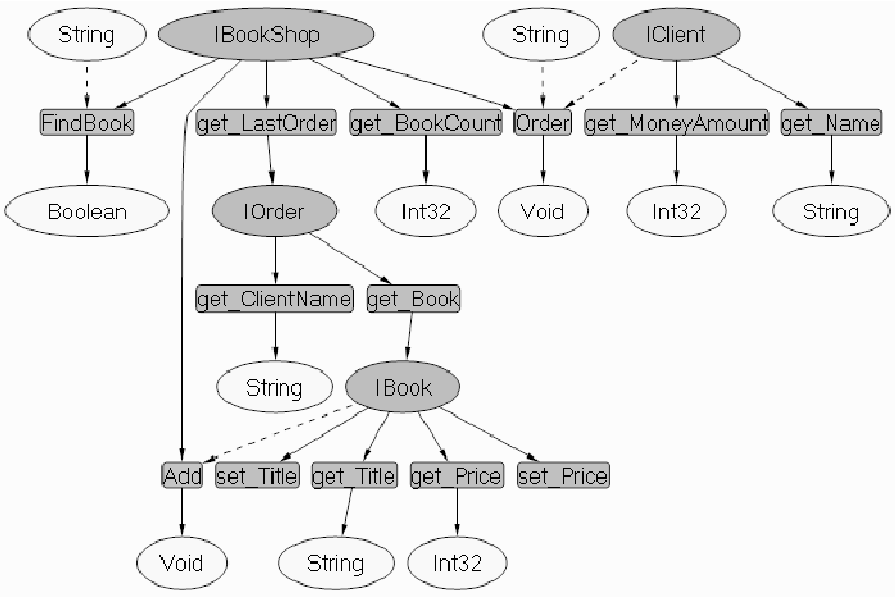

Рис. 2. Унифицированная модель API

На следующем этапе в модель добавляется дополнительная информация о параметрах интерфейсных функций и возможных значениях перезаписываемых свойств объектов. Выше было указано, что параметрами могут являться объекты, как простой, так и сложной структуры. Поскольку параметры сложной структуры задекларированы внутри интерфейса, их можно получить обратным ходом по графу модели путем вызова цепочки интерфейсных методов родительских объектов.

Для простых объектов можно определить домен значений, которые будут использоваться при генерации тестовых сценариев. В общем случае нам ничего не известно о том, как интерфейсный метод обрабатывает данные. В связи с этим тестовые сценарии должны строиться на основе стратегии «черного ящика». Наиболее перспективными кандидатами в такой домен выглядят значения, полученные при помощи методик разбиения по категориям и анализа граничных значений [4].

Было установлено, что для отдельных типов данных часть полученных таким образом значений повторяется независимо от семантики простого объекта. Например, для целых чисел «перспективными» в плане тестирования являются такие значения, как ноль, минимальное отрицательное число, максимальное отрицательное число, минимальное положительное число, максимальное положительное число. Эти значения могут быть заранее определены в шаблоне, а их добавление в

домен тестовых значений объекта будет происходить автоматически. Если существуют «перспективные» значения, зависящие от семантики объекта, они должны быть добавлены тестировщиком вручную. В данном случае невозможно извлечь знания о поведении интерфейсного метода из его спецификации, таким знанием обладает тестировщик. Например, для функции определения числа на простоту в домен могут быть добавлены типичное (не граничное) простое число и типичное составное число. Такое формирование тестовых доменов является важным этапом формализации функциональных требований.

Для представления функциональных требований используется известный подход на основе программных контрактов [5], состоящих из предусловий и постусловий интерфейсных методов и инвариантов типов данных. Основное его преимущество в том, что он позволяют автоматически построить оракулы и увеличить покрытие функциональности.

Рассмотренный в статье подход к автоматизации тестирования систем с интерфейсом программирования реализован в рамках программной среды APITest. APITest позволяет работать со спецификациями во всех рассмотренных выше форматах, а также непосредственно с динамическими библиотеками DLL. Производится синтаксический анализ спецификации и автоматическое построение унифицированной модели, представленной на рис. 2. Диаграммы UML в своем графическом представлении неудобны для анализа и трансляции в другие языки и модели. Чтобы избавиться от этого ограничения, необходимо представить диаграмму в текстовом виде при помощи формата XMI (XML-based Metadata Interchange).

APITest имеет графический интерфейс редактирования доменов значений простых объектов, а также интерфейс наложения предусловий и постусловий на объекты и операции. Для удобства и наглядности редактирование происходит с привязкой к модели интерфейса программирования.

В зависимости от используемого шаблона осуществляется запись тестовых сценариев в виде последовательности вызовов интерфейсных методов на конкретном языке программирования. Полученный таким образом код может быть выполнен в одной из сред программирования с использованием средств автоматизации выполнения тестов, например библиотек семейства xUnit.

Разработанная программная среда используется для проведения экспериментальных исследований для подтверждения работоспособности и эффективности предложенного метода. Проводятся исследования показателей повышения скорости и эффективности тестирования. Результаты экспериментов будут изложены в последующих статьях.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Мейерс Г.* Искусство тестирования. – М.: Финансы и статистика, 1982.
2. *Whittaker J.* Stochastic software testing // Annals of Software Engineering. Vol. 4. – P. 115-131. –1997.
3. *Бирюков С.В.* Об использовании моделей при автоматизации построения тестовых сценариев для тестирования интерфейса программирования приложения // Молодежь и современные информационные технологии. Сборник трудов VII Всероссийской НПК студентов, аспирантов и молодых ученых "Молодежь и современные информационные технологии", ч.2. –Томск: Изд-во СПБ Графикс. С.135-136.
4. *Бирюков С.В.* Анализ стратегий тестирования программного обеспечения // Известия ЮФУ. Технические науки. Специальный выпуск. – 2008. – № 1 (78). – С. 59-63.
5. *B. Meyer.* Applying `Design by Contract' // IEEE Computer, vol. 25, No. 10, October 1992. – P. 40-51.

Бирюков Сергей Вячеславович
Технологический институт федерального государственного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: birser@mail.ru.
347928, г. Таганрог, пер. Некрасовский, 44.
Тел.: 89287642928.

Birukov Sergey Vaycheslavovich
Taganrog Institute of Technology – Federal State-Owned Educational Establishment of Higher Vocational Education “Southern Federal University”.
E-mail: birser@mail.ru.
44, Nekrasovskiy, Taganrog, 347928, Russia.
Phone: 89287642928.

УДК 550.34.016

А.Е. Родионова

ОБРАБОТКА ИНФОРМАЦИИ В ЗАДАЧЕ ТЕПЛОВОГО МОНИТОРИНГА И КАРТИРОВАНИЯ МЕСТНОСТИ

Доклад посвящен задачам теплового (инфракрасного) мониторинга и картирования местности. Дан обзор типичных приложений, в которых возникают подобные задачи, рассмотрены характерные для данных задач трудности. Рассказано о программно-аппаратном комплексе, разработанном с участием сотрудников ИПУ РАН, и приведены примеры обработки системой реальных съемочных данных.

Аэросъемка; тепловое инфракрасное изображение; инерциальная система ориентации.

A.E. Rodionova

INFORMATION DATA PROCESSING IN THE TASK OF THERMAL MONITORING AND MAPPING

The report is dedicated to the tasks of monitoring and mapping in the thermal (infrared) spectral band. A review is given for the typical applications including difficulties faced while solving the relative tasks. Software together with the hardware created by, ICS RAS employees participation these tasks solution are introduced. Examples of the survey data processing are also given.

Airborne survey; thermal infrared image; inertial attitude system.

Введение. Усиливающееся негативное воздействие человека на все компоненты природной среды требует применения новых более эффективных средств и методов контроля. Поэтому в последнее время все более актуальной становится реализация систем мониторинга при оценке состояния природных и техногенных объектов. Задача, определившая направление данной работы, состояла в получении тепловых снимков по результатам инфракрасной (ИК) аэросъемки.

1. Области применения теплового мониторинга. ИК-съемка используется в ряде актуальных задач.

1. Определение местоположения и диагностика состояния продуктопроводов, в частности, подземных тепловых сетей, с выделением предаварийных и аварийных участков**.** Характерной чертой современного города является развитая система тепло- и водоснабжения. Нарушения в работе этой системы могут иметь самые разнообразные последствия – от изменения состава и температуры грунтовых вод,