дисциплине «Информационные технологии в экономике и управлении» на экономическом факультете РГГМУ. Результаты тестирования выявили тесную корреляцию между тестовыми экзаменационными оценками и результатами текущей аттестации успеваемости студентов.

Для ввода в КОСТ тестовых заданий не требуется высокая квалификация преподавателя в области информационных технологий – достаточно элементарных знаний в Excel на уровне средней школы.

КОСТ имеет открытый код, и поэтому любой пользователь, владеющий основами языка программирования VBA, может совершенствовать алгоритмы тестирования.

Достоинствами КОСТ являются:

1. Возможность ввода ответа, который может состоять из нескольких предложений практически в свободном виде.
2. Выполнение заданий с помощью приложений, установленных на персональном компьютере или сервере.
3. КОСТ может быть установлен на любом компьютере, оснащённом приложением MS Excel.
4. Студенты могут скопировать КОСТ вместе с тестовыми заданиями для домашней подготовки к тестированию.
5. Система не требует наличия Интернет или локальной сети.
6. Система может быть использована для одновременного тестирования любым количеством студентов в компьютерном классе.

Карпушинский А.М., Павловская Т.А.

Автоматизированная генерация тестов для объектно-ориентированных программ

(СПбГУ ИТМО, СПбГУЭФ, Санкт-Петербург)

При тестировании ПО часто требуется найти такие тестовые наборы, которые бы покрывали специфичные особенности программы. Отыскивать такие тесты вручную – процесс чрезвычайно долгий и трудоемкий, особенно когда программа сложная, поэтому в последнее время прилагаются усилия к созданию средств, автоматизирующих этот процесс, иначе: разработке методов автоматизированной генерации тестовых данных (АГТД). Зачастую желаемые входные данные должны соответствовать сложным ограничениям, поэтому простой метод случайной генерации становится недейственным. В противовес случайному методу, методы оптимизации (в том числе использующие генетические алгоритмы) предназначены для решения сложных проблем, связанных с одновременным удовлетворением многим ограничениям. Большинство существующих методов поиска тестовых данных ориентированы на структурные языки программирования, в то время как объектно-ориентированные языки (такие как Java, C# – далее ООЯП) гораздо более

сложны в тестировании. Инкапсуляция, наследование, полиморфизм, обработка исключений, события и прочие механизмы ООЯП привносят неявный поток управления.

Все ООЯП является событийно-управляемыми, то есть передача управления внутри программы осуществляется как прямо (непосредственный вызов одних функций из других), так и косвенно (посредством генерации *сообщений* одним объектом и передачи их другим объектам для обработки). Механизм обмена сообщениями характеризуется тем, что при выполнении программы контроль всегда возвращается вызывающему объекту, когда сообщение обработано. Основными источниками сообщений в ООЯП являются события (например, event в C#), асинхронные вызовы (которые, по сути, являются комбинацией событий) и исключения (exceptions).

Механизм обработки исключений представляет собой последовательность выброса исключения и его обработки в том же методе, в котором оно было выброшено, либо в одном из вызывающих (объемлющих) методов, находящихся ниже в стеке вызовов [1]. Таким образом, сообщение генерируется в том случае, если при выбросе исключения управление передается в один из вызывающих методов.

Учитывая вышеизложенное, тестирование программы, написанной на ООЯП, можно разбить на два уровня: модульное тестирование (при котором каждый класс тестируется отдельно (при этом используются методы структурного тестирования, применимые для «процедурных» языков), и интеграционный (необходима адаптация существующих методов АГТД к механизму обмена сообщениями в ООЯП).

На уровне модуля тестирование каждого класса начинается с построения управляющего графа каждого класса программы, выбора критерия покрытия структурных элементов программы (покрытие операндов, дуг или путей), а также разработки алгоритма нахождения входных тестовых данных, удовлетворяющих выбранному критерию. На интеграционном уровне каждая функция в классе является «черным ящиком», а механизм обмена сообщениями отражается диаграммой последовательностей, которая является формальной спецификацией программы. При этом осуществляется переход от структурной модели программы к ее поведенческой модели.

Целью статьи является разработка нового метода АГТД, использующего метод глобальной оптимизации на основе генетического алгоритма. Предлагаемый метод адаптирован для решения проблемы тестирования объектно-ориентированного кода, в частности, содержащего обработку исключений как одного из аспектов событийно-управляемой модели.

Обзор методов АГТД

В настоящее время известны следующие методы АГТД для программ процедурной парадигмы:

- Статические методы:
  - Символьное выполнение
  - Тестирование на основе ограничений

- Динамические методы:
  - Случайное тестирование
  - Локальный и глобальный поиск, а именно:
    - моделирование отжига;
    - метод Миллера-Спунера;
    - метод вариации переменной Корела;
    - цель-ориентированный метод;
    - генетические алгоритмы;
    - алгоритмы роевого интеллекта;
    - меметические алгоритмы – гибрид эволюционных алгоримов и алгоритмов локального поиска

Статические методы генерации (такие как метод символьного выполнения и метод ограничений) достаточно хорошо описаны и используются при тестировании процедурного кода. Они основаны на анализе внутренней структуры тестируемой программы и не требуют ее выполнения. В рамках статического метода производится обход заданного пути в управляющем графе и построение символьного представления значений внутренних переменных как выражений, содержащих только входные переменные.

Остальные перечисленные методы относятся к классу динамических. Они используют реальное выполнение программы, что позволяет избежать недостатков статических методов, а именно: неэффективности при наличии циклов, указателей и массивов (статический анализ зависимостей между входными данными и внутренними переменными затруднен), а в рамках ООЯП – при наличии неявного потока управления. К динамическим относится методы: случайного поиска, локального поиска, поиска с использованием методов глобальной оптимизации (симуляции отжига, генетических алгоритмов, алгоритмов роевого интеллекта).

Ниже представлено описание общего генетического алгоритма и рассмотрена его адаптация для использования в рамках проблемы АГТД при тестировании объектно-ориентированных программ.

Общие сведения о генетических алгоритмах

Генетический алгоритм (ГА) в общем случае является алгоритмом глобальной оптимизации некоторой целевой функции многих переменных. Он основан на процессах, подобных процессу эволюции в природе. Возможные решения задачи оптимизации именуются особями (хромосомами). Множество возможных решений образует популяцию. Каждая особь оценивается степенью приспособленности (целевой функцией, показывающей, насколько «хорошим» является решение в контексте данной задачи) [2, 3]. С помощью функции приспособленности среди всех особей популяции выделяют:

- наиболее приспособленные (более подходящие решения), которые получают возможность скрещиваться и давать потомство;
- наихудшие (плохие решения), которые удаляются из популяции и не дают потомства;

Таким образом, приспособленность нового поколения в среднем выше предыдущего. Возможные решения обычно кодируются (чаще всего, преобразуются в битовые строки) для применения к ним генетических операций. К этим операциям относятся скрещивание (кроссовер) и мутация.

В классическом ГА:

- начальная популяция формируется случайным образом;
- размер популяции (количество особей N) фиксируется и не изменяется в течение работы всего алгоритма;
- каждая особь генерируется как случайная L-битная строка, где L – длина кодировки особи;
- длина кодировки для всех особей одинакова.

На рисунке 1 изображена общая схема работы генетического алгоритма.

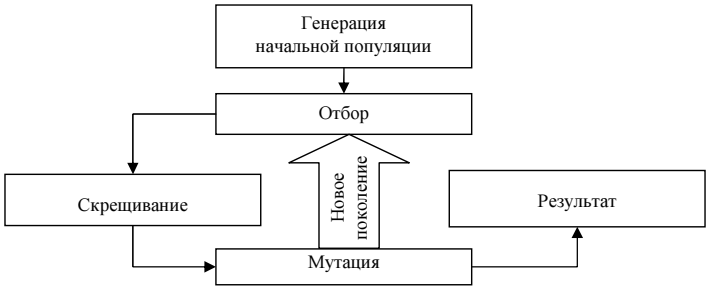

Рис. 1. Общая схема работы генетического алгоритма

Шаг алгоритма состоит из трех стадий:

1. Генерация промежуточной популяции путем отбора текущего поколения.
2. Скрещивание особей промежуточной популяции путем *кроссовера*, что приводит к формированию нового поколения.
3. Мутация нового поколения.

Промежуточная популяция – это набор особей, получивших право размножаться. Наиболее приспособленные особи могут быть записаны туда несколько раз, наименее приспособленные с большой вероятностью туда вообще не попадут. В классическом ГА вероятность каждой особи попасть в промежуточную популяцию пропорциональна ее приспособленности, т.е. работает *пропорциональный отбор*. Особи промежуточной популяции случайным образом разбиваются на пары, потом с некоторой вероятностью скрещиваются, в результате чего получаются два потомка, которые записываются в новое поколение, или не скрещиваются, тогда в новое поколение записывается сама пара. В классическом ГА применяется одноточечный оператор кроссовера: для родительских строк случайным образом выбирается точка раздела, потомки получаются путём обмена отсечёнными частями. К полученному в результате отбора и скрещивания новому поколению применяется оператор мутации, необходимый для «выбивания» популяции из локального экстремума и способствующий защите от преждевременной сходимости.

Такой процесс эволюции может продолжаться до бесконечности. Критерием останова может служить заданное количество поколений или *схождение* популяции. Схождением называется состояние популяции, когда все строки популяции находятся в области некоторого экстремума и почти одинаковы [3, 4]. То есть кроссовер практически никак не изменяет популяции, а мутирующие особи склонны вымирать, так как менее приспособлены. Таким образом, схождение популяции означает, что достигнуто решение, близкое к оптимальному. Итоговым решением задачи может служить наиболее приспособленная особь последнего поколения.

Применение генетического алгоритма для создания тестовых наборов

Рассмотрим ГА, адаптированный для решения проблемы интеграционного тестирования ООЯП. Положим, что каждый метод в отдельности протестирован и представляет собой «черный ящик», а передача управления между методами осуществляется посылками сообщений (будь то явный вызов или возврат, обработка события или исключения). Таким образом, программа представляется в виде совокупности классов, экземпляры которых «общаются» посредством посылки сообщений друг другу. Описание таких взаимодействий может быть отражено диаграммой последовательностей (рис. 2), и эта диаграмма может служить формальной спецификацией для разрабатываемой программы.

В любой момент времени класс может находиться в одном из своих состояний. Каждое состояние характеризуется совокупностью значений полей и наличием одного из полученных сообщений. Необходимо учитывать, что различные экземпляры одного и того же класса могут иметь отличное друг от друга поведение, то есть посылать различные сообщения и иметь, таким образом, разные множества состояний, которые могут пересекаться. В первую очередь это обусловлено возможностью определить для класса несколько конструкторов, в каждом из которых варьируются начальные значения полей.

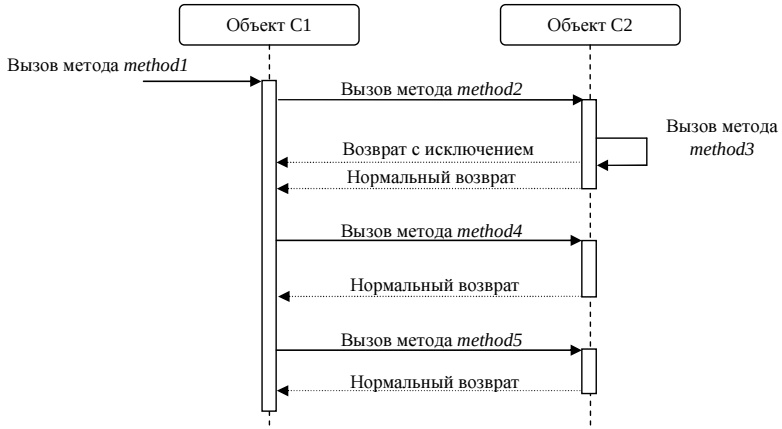

Рис. 2. Диаграмма последовательностей

Пусть дан класс C, и пусть S_C – множество состояний этого класса, а Φ_C – множество операций класса C, то есть множество возможных последовательностей передачи управления, доступных классу C. Тогда для каждой пары состояний s_1 и s_2 можно найти некоторое количество последовательностей операций Φ_C, которые обусловливают переход из s_1 в s_2 и, потенциально, могут выявить ошибку в классе C. Последовательность операций $\varphi \in \Phi_C$ есть функция отображения S_C на S_C. Таким образом, для различных начальных состояний данная последовательность может выявить ошибки в реализации.

Исходя из вышесказанного, тестовый набор при тестировании класса должен состоять из набора тщательно выбранных последовательностей переходов, а также набора состояний класса.

При интеграционном тестировании программы, состоящей из нескольких классов, формальной спецификацией может быть диаграмма последовательностей. При этом критерием тестирования является покрытие всех дуг (то есть всех возможных передач управления между методами всех классов). Иными словами, для диаграммы последовательностей S и тестируемой программы P тестовый набор T удовлетворяет критерию покрытия дуг, если каждая дуга в S, представляющая передачу сообщения, пройдена хотя бы один раз при выполнении P на данном тестовом наборе T. В общем виде необходимо, чтобы каждое сообщение в диаграмме было послано хотя бы один раз. На рисунке 3 представлена общая схема предлагаемого генетического алгоритма.

Пусть искомые тестовые наборы полагаются особями в рамках алгоритма. Хромосомой особи будем считать последовательность конструктора, одного или нескольких вызовов других методов (включая значения параметров) и посылки сообщений, а также номера состояния, в которое переходит класс в результате выполнения этой последовательности.

Первым шагом алгоритма является выделение списка всех дуг, которые нужно покрыть тестовыми наборами. Функцию приспособленности в данном случае определим как отношение количества дуг, приведших к дуге E (см. рис. 3) на текущем тестовом наборе к общему количеству дуг, которые могут привести к этой дуге. Мутации особей можно проводить одним из следующих способов:

- Изменение входных значений.
- Изменение конструктора.
- Добавление в последовательность вызова метода.
- Удаление вызова метода из последовательности.

Скрещивание двух выбранных с определенной вероятностью особей происходит оператором одноточечного кроссовера: случайным образом выбирается точка раздела, потомки получаются путём обмена отсечёнными частями. Критерием останова является схождение популяции либо установленное количество поколений. Увеличивая время работы алгоритма, можно добиться повышения эффективности алгоритма, однако бесконечное повышение максимального количества популяций не приводит к максимизации эффективности, к тому же, варьируя параметрами, необходимо избегать за-

стревания в локальных максимумах. На сегодняшний день продолжается исследование в области увеличения эффективности алгоритма, рассматриваются возможности модификации и применения алгоритма роевого интеллекта, а также многоточечного кроссовера.

Рис. 3. Алгоритм генерации тестовых данных

Реализация алгоритма

Предложенный алгоритм является частью разработанной системы АГТД, применяемой для тестирования программ, написанных с использованием ООЯП Java и содержащих обработку исключений. Экспериментальная разработка построена на языке C# .NET (.NET Framework v4.0) и включает следующие компоненты:

1. Синтаксический анализатор исходного кода на языке Java на базе свободно распространяемого инструмента antlr (www.antlr.org).
2. Построитель управляющего графа, использущий синтаксическое дерево исходной Java-программы и разработанный алгоритм разбора синтаксического дерева, учитывающий конструкции обработки исключений, для генерации внутреннего, а также текстового представления управляющего графа.
3. Визуализатор управляющего графа, позволяющий наглядно представить тестируемую программу в виде белого ящика, а также проследить все передачи управления в ходе ее выполнения.
4. Иструментатор, используемый для привнесения в текст исходной программы вызовов служебных процедур, фиксирующих ход выполнения программы.
5. Модули генерации тестовых данных, реализующие предложенный генетический алгоритм, а также уже существующие алгоритмы поиска тестовых данных, включенных с целью оценки эффективности разработанного алгоритма.
6. Среда компиляции, включающая компилятор java (java.exe) и среду многократного выполнения инструментированной программы на разных входных данных, функционирующая на основе командных файлов.

Работа системы на тестовых программах небольшого размера (до 10 методов и 200 строк) показала результаты, приемлемые по производительности. Эффективность поиска сравнима с работой систем, основанных на уже существующих динамических методах, таких как метод локального поиска Б. Корела и метод последовательной релаксации (без учета конструкций обработки исключений). В ходе апробации системы получаемые тестовые наборы почти всегда удовлетворяли критерию покрытия путей (в зависимости от количества сложных передач управления внутри методов). В настоящее время выполняется экспериментальная проверка границ и условия применимости разработанного алгоритма.

Заключение

В статье рассмотрены проблемы тестирования программ, написанных на объектно-ориентированных языках программирования, предложен способ автоматизированной генерации тестовых данных для тестирования программ, содержащих обработку исключений, разработан и апробирован метод генерации тестовых наборов, основанный на генетическом алгоритме.

Литература

1. Shujuan Jiang, Yuanpeng Jiang. An analysis approach for testing exception handling programs // SIGPLAN Notices. «ACM» – NY, 2007. – Vol. 42.
2. Sandhu P.S., Dhiman S.K., Goyal A. A genetic algorithm based classification approach for finding fault prone classes // World Academy of Science, Engineering and Technology. – 2009. – Vol. 60. – P. 485-488.
3. Буздалов М.В. Генерация тестов для олимпиадных задач по программированию с использованием генетических алгоритмов // Научно-технический вестник. – 2011. – № 2(72). – С. 72-76.
4. Новосельский В.Б., Павловская Т.А. Выбор и обоснование критерия эффективности при проектировании распределенных баз данных // Научно-технический вестник. – 2009. – № 2(60). – С. 76-82.

Соловьев Т.Г.

Потребители образовательных услуг высшего учебного заведения: идентификация и сегментация

(СарФТИ НИЯУ МИФИ, г. Саров)

Рост конкуренции в сфере образования, ужесточение требований со стороны потребителей к качеству образовательных услуг, демографический кризис, вхождение России в Европейское образовательное пространство привели к тому, что в настоящее время ориентация на потребителя является ключевым принципом и основной целевой установкой организации образовательной деятельности. Реализация данного принципа требует формирования системы устойчивого взаимодействия вуза с потребителями. Изучение потребителей образовательных услуг, их потребностей и ожиданий позволяют высшему учебному заведению устанавливать, выстраивать и развивать эффективные отношения с ними. Это в свою очередь формирует предпосылки повышения конкурентоспособности вуза на рынках образовательных услуг и труда. Установление долгосрочных доверительных отношений с потребителями способствует формированию базы лояльных, то есть приверженных вузу потребителей. Кроме того, благодаря длительным взаимоотношениям с потребителями у вуза формируется положительный имидж. Следовательно, каждый вуз должен идентифицировать имеющихся и потенциальных потребителей, умело сегментировать клиентскую базу. Решение этих задач позволяет оценить размеры и структурировать спрос на образовательные услуги, сформировать целевые рынки и адекватные им стратегии позиционирования.

Исследование практики реализации процесса идентификации потребителей образовательных услуг в нескольких российских высших учебных заведениях: Мордовском государственном университете им. Н.П. Огарева