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.

РАЗРАБОТКА ТЕСТОВ ДЛЯ ПРОГРАММ С НЕЯВНЫМ ПОТОКОМ УПРАВЛЕНИЯ

(СПбГУ ИТМО, Санкт-Петербург)

Автоматизированная генерация тестовых данных (АГТД) для программ позволяет сократить время разработки тестов и обеспечивает высокий процент покрытия ошибок. Объектно-ориентированные языки программирования (ООЯП) более сложны в тестировании, чем процедурные, поскольку наследование, полиморфизм, обработка исключений, события и прочие механизмы ООЯП привносят неявный поток управления, сильно усложняющий алгоритмы. Неявный поток управления можно представить как обмен сообщениями, характеризующийся тем, что после обработки сообщения при выполнении программы контроль всегда возвращается вызывающему объекту.

В данной работе рассматривается разработка тестов для объектно-ориентированных программ, содержащих один из наиболее распространенных видов неявного потока управления – обработку исключительных ситуаций (исключений). 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]. Сообщение генерируется в том случае, если при выбросе исключения управление передается в один из вызывающих методов.

Предлагается разбить тестирование программы, написанной на ООЯП, на два уровня: модульный, при котором каждый класс тестируется отдельно (при этом используются методы структурного тестирования, применимые для «процедурных» языков), и интеграционный, для которого применяется адаптация существующих методов АГТД.

На нижнем уровне для тестирования каждого класса строится управляющий граф, выбирается критерий покрытия структурных элементов программы (операндов, дуг или путей) и разрабатывается алгоритм нахождения 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.

В данной статье описан новый метод АГТД, использующий глобальную оптимизацию на основе генетического алгоритма, для объектно-ориентированного кода, содержащего обработку исключительных ситуаций.

На рис. 1 изображена 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.

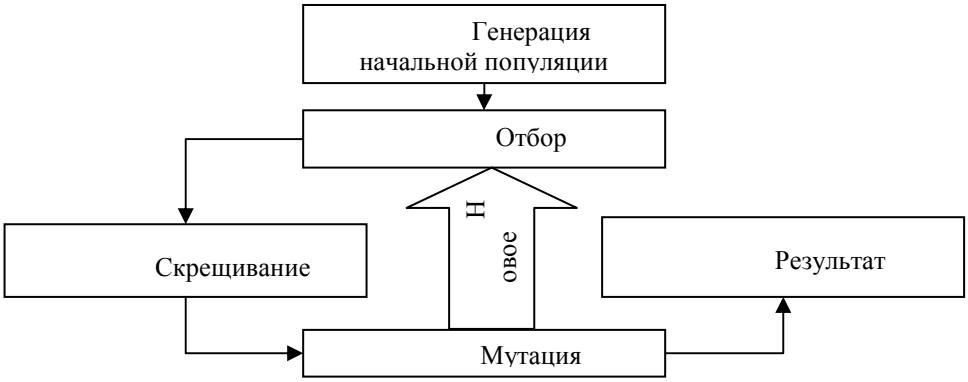

Рис. 1. Общая схема работы генетического алгоритма

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 применение ГА для создания тестов интеграционного тестирования ООЯП. Пусть каждый метод в отдельности протестирован и представляет собой «черный ящик», а программа представлена в виде совокупности классов, экземпляры которых взаимодействуют путем посылки сообщений. Описание таких взаимодействий может быть отражено диаграммой последовательностей (рис. 2), которая может служить формальной спецификацией для разрабатываемой программы.

В любой момент времени класс может находится в одном из своих состояний. Каждое состояние характеризуется совокупностью значений полей и наличием одного из полученных сообщений. Различные экземпляры 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.

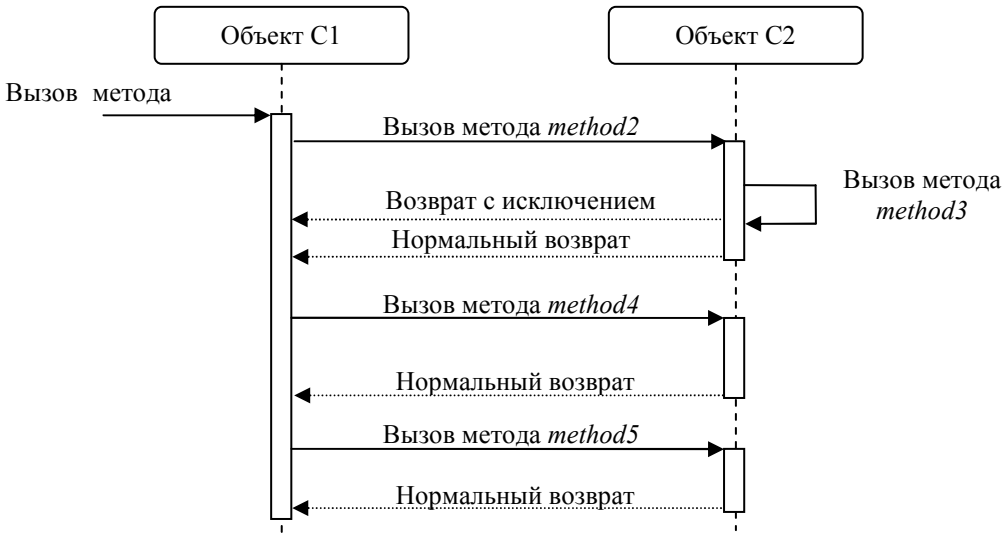

Рис. 2. Диаграмма последовательностей

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 обуславливают 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. Следовательно, тестовый набор для класса должен состоять из набора тщательно выбранных последовательностей переходов, а также набора состояний класса.

Критерием тестирования программы 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 рис. 3 представлена общая схема предлагаемого генетического алгоритма.

В качестве особей в ГА выступают искомые тестовые наборы. Хромосомой особи будем считать последовательность из 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.

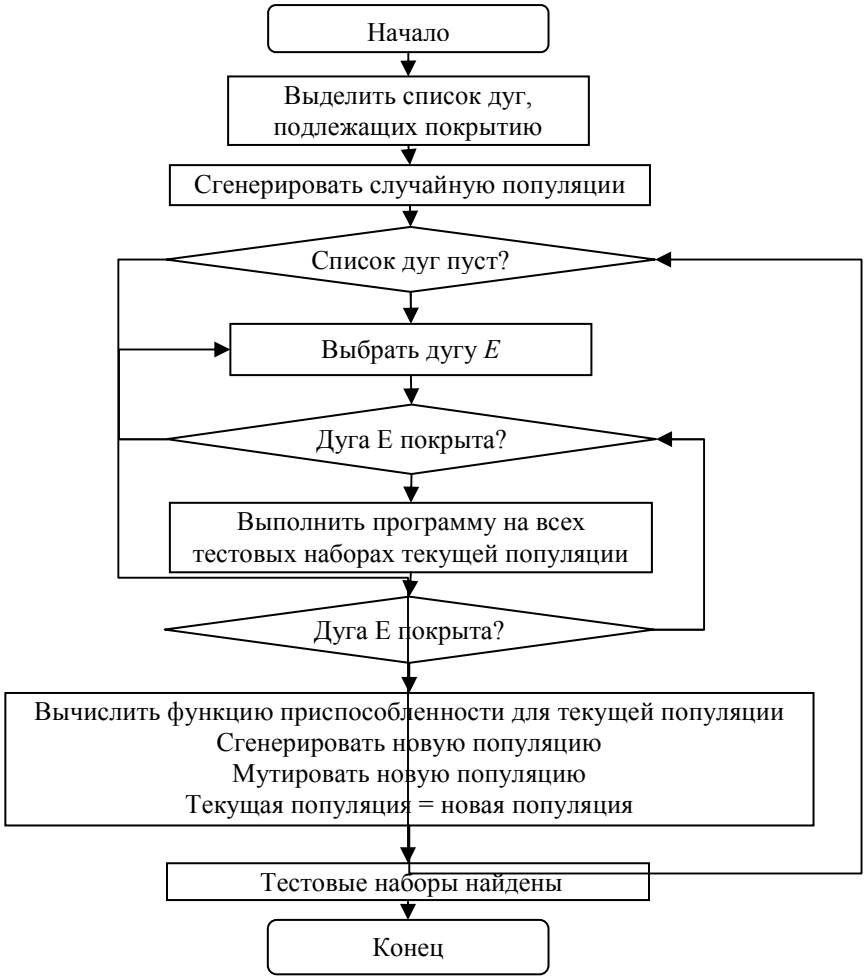

Рис. 3. Алгоритм генерации тестовых данных

Первым шагом алгоритма является выделение списка всех дуг, которые нужно покрыть тестовыми наборами. Функцию приспособленности 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.

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 генерирующий внутреннее, а также текстовое представление управляющего графа.
3. Визуализатор управляющего графа.
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 данной работе рассмотрены проблемы тестирования ООЯП, предложен способ АГТД объектно-ориентированных 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, USA. 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. – № 02(72). – 2011. – С. 72-76.
4. Новосельский В.Б., Павловская Т.А. Выбор и обоснование критерия эффективности при проектировании распределенных баз данных // Научно-технический вестник. – № 02(60). – 2009. – С. 76-82.

Смирнов А.И.

ПРОГНОЗИРОВАНИЕ ЗНАНИЙ СТУДЕНТОВ НА ОСНОВЕ МОДЕЛИ Г. РАША

(УИЭУиП, Екатеринбург)

В настоящее время в связи с проводимыми в сфере образования реформами вопросы контроля и управления качеством обучения приобретают большое значение. Широкое распространение в качестве методов контроля знаний получили, в частности, различные процедуры тестирования. Вместе с тем, при использовании результатов тестирования вузы часто ограничиваются простейшей статистической информацией, не проводя глубокий их анализ математическими методами. Вузы, использующие математические методы анализа, в свою очередь, как правило, ограничиваются оценкой эффективности теста в целом, а также его корректности, уровня сложности и согласованности отдельных тестовых заданий (конструирование, анализ и модернизация теста [1], [2], [3]). Разработка же математических моделей, позволяющих предсказывать уровень знаний студентов после обучения (и соответственно результатов их тестирования), не нашла столь широкого распространения ([4]).

Между тем предсказывающие модели могли бы использоваться в процедурах управления и оптимизации учебным процессом. При обработке результатов тестирования широко используется однопараметрическая модель Г. Раша, позволяющая разделить влияние на конечные результаты тестирования уровня знаний студентов и сложности тестовых заданий ([2]). После тестирования группы студентов с помощью полученной матрицы тестирования можно вычислить значения уровня сложности каждого из заданий и уровня знаний каждого из студентов. Поскольку считается, что уровень знаний студентов не зависит от проведенного теста, можно предсказать результаты прохождения теми же студентами другого теста, для которого известны уровни сложности зада-