XXXI Неделя науки СПбГПУ. Материалы межвузовской научной конференции. Ч. VI: С.38-39, 2003.

УДК 658.512.011.56: 681.3.06

Е.М. Вдовец (асп., каф. ИУС), П.Д. Дробинцев (6 курс, каф. ИУС),
В.П. Котляров, к.т.н., проф.

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ВЕРИФИКАТОРА ФУНКЦИОНАЛЬНЫХ СПЕЦИФИКАЦИЙ

Данная статья посвящена вопросам автоматизации тестирования верификатора функциональных спецификаций. Так же, данный подход можно использовать для автоматизации приложений с интерфейсом командной строки, работающих с MSC диаграммами.

Задача состоит в создании тестовых наборов и автоматизации тестирования консольного приложения, работающего под управлением операционной системы Unix. Технически задача делится на две части – непосредственное создание тестовых наборов и автоматизация процесса принятия решения о соответствии поведения тестируемого приложения требованиям функциональной спецификации.

Особенности и предназначение верификатора функциональных спецификации.В настоящее время большое внимание уделяется возможности генерации как тестов, так и исходного продукта. Одной из активно развивающихся областей является генерация кода из функциональных спецификаций, написанных на языках высокого уровня, таких как UML, SDL, MSC. В связи с этим возникает проблема проверки правильности написания подобных спецификаций, имеются в виду возможная неполнота описания состояний системы, ошибки связанные с использованием временных спецификаций и другие. Тестируемый верификатор позволяет избежать подобных ошибок путём выявления конфликтных мест в спецификациях, написанных с использованием языка MSC. MSC – Message Sequence Charts – диаграммы последовательности сообщений (стандарт ITU z.120) – язык, предназначенный для спецификации и описания взаимодействия компонент системы и их окружения посредством обмена сообщениями. По окончанию своей работы верификатор выдаёт графический и текстовый вердикт по трём составляющим:

Временная непротиворечивость - позволяет определить соответствие структуры диаграммы и её временных спецификаций.Непротиворечивость переходов - позволяет определить места возникновения недетерминированного поведения системы.

Полнота - позволяет определить полноту описания системы набором MSC диаграмм.Полученный вердикт может использоваться разработчиком для коррекции созданных спецификаций, что приводит к уменьшению количества возможных ошибок в конечном продукте.

Анализ и разработка системы тестирования верификатора функциональных спецификаций. При тестировании верификатора возникла необходимость получения сценариев, написанных на языке MSC. В качестве доступных вариантов решения были рассмотрены две возможности: написание сценариев работы абстрактной системы, генерация тестовых сценариев на основе работы реальной системы

Первый вариант отличается большим объёмом ручной работы и меньшим количеством полученных сценариев, но позволяет написать сценарии нестандартного поведения, которые редко встречаются в работе реальных систем. Описанные свойства данного подхода делают его очень привлекательным для проведения так называемого критического тестирования, когда на вход системы подаются редко встречающиеся наборы входных данных, которые позволяют обнаружить ошибки, пропущенные на стадии обычного тестирования.

Второй подход даёт возможность получить большее количество сценариев и минимизировать затраты на их написание, а также получить сценарии наиболее приближенные к работе реальной системы. Таким образом, наиболее эффективным является подход, связанный

с генерацией большинства тестов и дополнением этого множества тестами, связанными со специфическим поведением описываемой системы.

Для тестирования верификатора был разработан модуль, подключаемый к модели операционной системы и позволяющий получать на выходе описание поведения модели на языке MSC. Данный модуль на каждом шаге работы модели сохраняет на диаграмме:

- состояние системы в виде условий с параметрами,
- вызовы сервисов ОС в виде входящих в модель сообщений,
- отклики модели в виде исходящих сообщений,
- конфигурацию модели.

При данном подходе модель используется как генератор эталонных сценариев поведения, совокупность которых представляет собой функциональную спецификацию системы. Возможность генерации обеспечивается заданием обобщенного сценария, определяющего начальное и конечное состояния системы, а также её конфигурацию; используя эти данные, модель отрабатывает все возможные варианты поведения, а также пути достижения конечного состояния.

На этапе разработки системы автоматизации тестирования в модель была добавлена возможность определения текущего состояния, применяемая в дальнейшем для тестирования свойств верификатора, связанных с поиском недетерминированного поведения, описанного в функциональных спецификациях. Полученные таким образом сценарии содержат информацию о состоянии системы на каждом шаге.

Для успешного решения задач, поставленных перед системой автоматизации тестирования, была реализована система скриптов, которые принимают на вход тестовые данные, запускают тестируемое приложение и сравнивают полученный результат с ожидаемым. Это означает, что ожидаемый выход приложения должен быть представлен в неком стандартном виде, причем для упрощения процедуры внесения изменений все ожидаемые выходы запускаемых тестов должны находиться в едином файле. Для разрешения этой проблемы был разработан специальный язык, на котором и был написан так называемый table-файл.

Каждая строка table-файла имеет следующую структуру:

<имя теста> <описание ожидаемого выхода>

здесь <имя теста> - имя каталога, где находятся входные данные

<описание ожидаемого выхода> - конструкция специального table-языка.

Table-язык содержит следующие конструкции:

- <file_name> <message> (поиск в заданном файле <file_name> сообщения с текстом <message>)
- etalon_<file_name> (сравнение файла <file_name> с эталонным)
- <file_name> contains_no <message> (заданный файл <file_name> не должен содержать сообщения с текстом <message>)
- <file_name> file_exists (файл <file_name> должен существовать)
- <file_name> file_not_exists (файл <file_name> не должен существовать)

С помощью этих конструкций можно записать ожидаемый выход тестируемого приложения и указать проверяющему скрипту на необходимую последовательность проверок. Таким образом, на выходе системы тестировщик должен будет только посмотреть результаты прохождения тестов в специальном log-файле.

За счёт предложенного подхода достигается значительное уменьшение времени, необходимого для проведения тестового цикла, а также упрощается анализ результатов тестирования.

Выводы. В статье предложен подход к автоматизации тестирования верификатора функциональных спецификаций с использованием модели реальной системы. На основании полученных в процессе реального использования результатов можно сделать выводы о целесообразности использования подобных методов в автоматизации тестирования.