УДК 681.3

Н.В.Воинов (асп., каф. ИУС), А.О.Веселов (5 курс, каф. ИУС), В.П.Котляров, к.т.н., проф.

ТЕХНОЛОГИЯ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ UML ПРОЕКТОВ

В настоящее время определяющим фактором разработки качественного программного продукта является эффективный дизайн и сбалансированность архитектурных решений. Традиционный подход на основе субъективных решений в дизайне, принимаемых даже талантливыми исполнителями, не может гарантировать такие свойства программного продукта, как полнота, корректность, безопасность или фиксированный уровень плотности дефектов. Именно поэтому в практике индустриального программирования все чаще используются формальные нотации и формальные методы контроля свойств программного продукта. Среди наиболее популярных формальных нотаций выделяется язык UML (Unified Modeling Language) [1], в котором интегрировано большинство изобразительных возможностей современных формальных спецификаций.

Единственно достоверной спецификацией является UML модель, специфицирующая функциональность в ее текущем состоянии. Преимущественной формой представления подобной поведенческой модели является диаграмма State Machine (SM) языка UML. Предлагается использовать этот тип моделей и на основе анализа диаграмм SM выводить поведенческие сценарии, необходимые для создания тестовых сценариев, и покрывающие всю разработанную на текущий момент функциональность. Такие сценарии позволяют, во-первых, проверить и подтвердить непротиворечивость решений, принятых в дизайне продукта, а также получить и отладить тестовый набор, пригодный к исполнению на модели, макете или промежуточной версии разрабатываемого продукта.

Полученные сценарии, использованные в тестировании, позволяют контролировать лишь ту функциональность, которая уже разработана, но зато такие сценарии могут служить основой для сравнения совпадения релиза с детальной спецификацией, выведенной из требований на более позднем этапе. Кроме того, на основе переиспользования фрагментов сценариев можно воспроизвести другие, например, контролирующие режимы использования продукта, обработку ошибочных ситуаций, выдачу предупреждающей диагностики и т.д. На определенном этапе зрелости технологии дизайна можно генерировать набор сценариев, полностью покрывающий все пункты требований в соответствии с заданным критерием.

Рассматриваемая технология предназначена для автоматической генерации тестов из UML SM диаграмм сначала на промежуточный язык сценариев (UML Sequence диаграммы или MSC), а затем на стандартный языке тестирования телекоммуникационных приложений - TTCN (Testing and Test Control Notation). Предлагаемый подход позволяет исключить ручную разработку и сосредоточить усилия тестировщиков исключительно на сценариях тестирования, что ускоряет процесс тестирования и фиксации ошибок.

Технология поддерживается системой скриптов и шаблонов автоматической генерации результирующих TTCN-файлов, использующих вспомогательные файлы описания типов данных, описания шаблонов сигналов, описания конфигураций т.п.

Технологическая цепочка базируется на использовании двух входных файлов. Создание данных файлов является самым трудоёмким, но может быть осуществлено на фазе разработки формальных спецификаций:

1. Непосредственно UML-диаграммы с сигналами и их параметрами. В диаграмме возможно использование различных конструкций языка UML, в том числе нелинейных – альтернатив, циклов, ссылок на другие диаграммы и т.д.
2. XLS-файл, где перечислены значения параметров сигналов, представленных на диаграмме.

Процесс генерации тестовых сценариев включает в себя несколько этапов. Все этапы осуществляются автоматически и тестировщик рассматривает данные этапы как единый процесс.

- На первом этапе осуществляется преобразование диаграммы из формата UML в формат MSC (Message Sequence Charts).
- Второй этап обеспечивает генерацию файлов с описанием функций слоя wrapper (“обертка”), которые выполняют преобразование модельных сигналов диаграммы в вызовы функций.
- На третьем этапе происходит генерация вспомогательного файла описания сценария в виде текстового представления сценария в формате MSC.
- Четвертый этап обеспечивает подготовку тестового прогона. На нем определяются конкретные значения параметров тестирующего сценария. Значения берутся из XLS-файла, разработанного тестировщиком.
- Пятый этап обеспечивает непосредственную генерацию тестового набора, ориентированную на исполнение на целевой платформе или ее симуляторе.
- Последний, шестой, этап выполняет генерацию файла запуска тестового цикла.

После автоматического выполнения шести шагов технологической цепочки все полученные файлы фиксируются в тестовом проекте. После компиляции и сборки проекта возможен прогон тестов в автоматическом режиме. Результаты тестирования сохраняются в Log-файле.

Многочисленные эксперименты позволили оценить время прохождения всей цепочки при наличии готовых UML-сценариев в несколько минут, что значительно меньше времени ручной разработки TTCN-тестов. Даже учитывая затраты, необходимые на разработку UML-диаграммы (в случае отсутствия готовой спецификации тестового сценария) и XLS-файла для настройки конкретного тестового прогона, выигрыш по времени на получение только одного TTCN-теста в соответствии с предложенной технологией составляет приблизительно несколько часов.

Кроме того, в случае изменения исходных требований или добавления в проект новых, процесс изменения тестовых сценариев сводится лишь к изменению исходных UML-диаграмм. Что в случае использования диаграмм требует минимальных затрат и соответствует добавлению некоторых сигналов в исходные диаграммы. Дальнейшая перегенерация всего тестового набора занимает всего несколько минут, что существенно меньше, чем ручное исправление кода тестов с возможностью внести дополнительные ошибки. К сожалению, в настоящее время изменение требований во время реализации проекта, явление весьма распространённое. Это обуславливается в основном существенными размерами проектов. Поэтому данная особенность разработанной технологической цепочки резко повышает её привлекательность.

Отсюда использование технологии генерации тестов из UML SM-моделей дает ощутимые преимущества для промышленных программных проектов в телекоммуникационных приложениях, для которых характерны большие объемы тестирования, большая трудоемкость разработки огромного количества тестовых наборов, большой размер отдельных тестовых сценариев по сравнению с технологией ручной разработки тестов.

ЛИТЕРАТУРА:

1. OMG. UML 2.0 Infrastructure Specification. September, 2004. http://www.omg.org.