XXXII Неделя науки СПбГПУ. Материалы межвузовской научно-технической конференции. Ч.V : С.22-23

УДК 658.512.011.56: 681.3.06

В.А.Сухомлинов (асп., каф. ИУС), В.П.Котляров, к.т.н., проф.

НАСТРАИВАИМАЯ ГЕНЕРАЦИЯ ТЕСТОВ ПО ФОРМАЛЬНЫМ СПЕЦИФИКАЦИЯМ

Формальные спецификации программных систем получают все большее распространение в практике разработки коммерческих продуктов. Существующий на рынке инструментарий часто не удовлетворяет потребностей в гибкости входного языка и возможностей настройки на целевую платформу. Формальные спецификации, использующие в основе графическое представление (такие как MSC, SDL, UML), сочетают в себе наглядность и формализм, благодаря чему получили широкое распространение в области телекоммуникаций и распределенных систем.

Целью данной работы является реализация концепции модульной системы, допускающей расширение пользователем. Ядро системы представлено сервисными функциями работы с операционной системой и базовой функциональностью, поддерживающей фундаментальные для системы понятия диаграмм последовательностей сообщений (MSC), спецификаций частичного порядка, дерева поведения (ДП) и конечный автомат (КА).

Пользователь имеет возможность настроить входной язык, разработав собственный разборщик языка, заполняющий стандартные структуры данных. В зависимости от класса входного языка (MSC, частичный порядок или КА) и желаемого типа выходного языка ядро системы осуществляет преобразование спецификаций. Различные параметры и стратегии преобразований могут меняться пользователем. В реализованной системе имеются штатные разборщики для языка MSC’2000 (рекомендация ITU z.120 11/99) и простого MSC-подобного языка. С другой стороны, пользователь реализует собственный модуль генерации кода или использует штатный модуль для структуры данных. В случае использования штатного модуля генерируется не готовый код, а шаблон на языке Tcl/Tk. Шаблон оперирует командами вида «послать сообщение», «принять сообщение» и т.п. Реализация этих команд необходима для генерации целевого кода. Обычно переиспользование имеющегося модуля требует меньше затрат, чем разработка нового, но не всегда обеспечивает необходимую гибкость. В настоящий момент реализованы шаблоны для генерации кода на C, C# и Verilog.

При использовании штатных средств настройка генерации тестового агента к системе осуществляется на 3-х уровнях: определяется способ включения (тип выходного языка: пассивный/активный, КА/ДП/и т.п.); описывается, как осуществляется взаимодействие с тестируемой системой и анализ результата (шаблон); реализуется адаптер между интерфейсами тестируемой системы и тестового агента.

Для поддержки работы тестового агента разработана библиотека, реализующая ядро КА и осуществляющая протоколирование результатов работы. Благодаря возможности подключать внешние модули протоколирования результатов исполнения возможна интеграция с коммерческими системами визуализации результатов, как, например, редактор wave-диаграмм Cadence SimLink. Библиотека также поддерживает работы параллельных КА, что позволяет во многих случаях избежать комбинаторного взрыва при моделировании асинхронных взаимодействий.

Сгенерированный тестовый агент способен проверять альтернативные последовательности воздействий, параметры сигналов, временные ограничения, состояния системы. Возможность параметризации тестовых сценариев в сочетании с высокой скоростью исполнения позволяют увеличить количество исполненных тестов, повысить

покрытие кода системы, увеличить вероятность нахождения дефектов. Использование на входе формального языка (MSC) делает тесты наглядными и предлагает больше возможностей для их последующего повторного использования. Время, необходимое на разработку тестов, снижается от 1.5 раза при первом применении в предметной области до 10 раз при повторном использовании.

Описанный подход был опробован в области телекоммуникаций и моделирования-симуляции электронных устройств. В результате были найдены дополнительные дефекты, пропущенные при классическом тестировании.

Таким образом, применение подхода настраиваемой генерации для тестирования демонстрирует его преимущество перед традиционными подходами, позволяет сократить время на разработку тестов и проведение тестирования, что, в свою очередь, способствует сокращению цикла разработки качественного программного продукта.