Апробация технологии тестирования UniTesK

Д.В.Кознов Н.А.Арчак
dim@tepkom.ru nicka@oktet.ru

Санкт-Петербургский государственный университет
198504, Университетский пр., 28
Санкт-Петербург, Россия

Аннотация

В данной работе рассматривается проект по независимой апробации технологии тестирования UniTesK. Цель апробации — дать оценку того, насколько авторам технологии удалось формализовать свой опыт в области тестирования и сделать его переиспользуемым. Актуальность этого исследования связана с тем, что UniTesK превращается из технологии, используемой только авторами, в "коробочный" продукт.

Введение

Проблемы тиражирования успешного опыта, независимо от предметной области, далеко не всегда решаются успешно. Часто оказывается, что коллектив опытных специалистов, успешно работающий в данной области много лет, тем не менее не может формализовать хотя бы часть своего опыта так, чтобы была возможность его повторного использования другими специалистами в этой же предметной области. Или же использование технологии требует серьезного обучения перед началом ее использования, или необходим большой объем работы по ее настройке. Бывает и так, что выявляются ограничения для случаев, с которыми авторы не встречались или для особенностей, которые не им удалось "упаковать" в технологию.

Группа RedVerst (Research and Development Verification, Specification, Testing) Института системного программирования

РАН много лет работала в области тестирования программного обеспечения, успешно выполнив ряд промышленных проектов по тестированию ПО для компаний Nortel Networks, Microsoft. В результате группа RedVerst создала технологию тестирования UniTesK [2], которую сейчас оформляет в виде "коробочного" продукта.

Исследовательский проект, представленный в этой работе, был направлен на независимую апробацию технологии UniTesK. Цель апробации — дать оценку того, насколько удачно обобщен многолетний опыт группы RedVerst, можно ли им пользоваться а) независимо от группы RedVrest, б) в новых предметных областях (т.е. там, где у авторов технологии нет промышленного опыта).

В проекте использовался продукт CTesK-Lite — вариант технологии UniTesK для создания тестового окружения в среде Microsoft Visual C++.

В качестве материала для апробации было выбрано ПО, реализующее IPv6-протокол (такое ПО уже тестировалось группой RedVerst), предоставленное компанией Октет, и DSP-система по обработке звука, разработанная компанией ЗАО ЛАНИТ-ТЕРКОМ. Последний тип приложения еще не тестировлся с помощью технологии UniTesK. Апробация проводилась на кафедре системного программирования СПбГУ группой, состоящей из четырех человек: один тестеровщик, два автора-консультанта DSP-системы, организатор эксперимента. Апробации предшествовал недельный курс обучения, проведенный группой RedVerst.

Проект по апробации столкнулся с рядом проблем, его результаты, будучи в общем положительными, тем не менее не однозначны. Простота технологии и ее надежность как программного продукта контрастировали со сложностью создания на основе CTesK-Lite конкретного решения по тестированию.

1 Контекст апробации

Тиражирование технологических решений в программировании. При создании и эксплуатации программно-организационных решений, созданных для преодоления конкретных проблем определенного проекта по разработке про-

граммного обеспечения (будем далее называть такие решения технологическими решениями), часто возникает потребность переиспользовать полученные результаты. Это можно делать:

1. для дальнейшего использования решения представителями заказчика отдельно от группы его авторов (в частности, если авторы являются представителями другой организации);
2. для реализации подобных решений для других проектов и заказчиков (либо в форме предоставления заказчику непосредственных результатов работы решения — например, найденных ошибок, если речь идет о решении по тестированию, либо передача в эксплуатацию самого решения); фактически, этот способ переиспользования означает рождение из решения технологии, которая пока неотделима от ее создателей;
3. для реализации технологии в виде "коробочного" продукта, что означает, что технологию удается отделить от ее создателей — от их уникального опыта, знаний, образования и т.д. — и сделать доступной широкому кругу пользователей.

Представленное выше можно рассматривать как сценарий (по всей видимости, не единственный) рождения "коробочной" технологии. При ее создании нужно тщательно отделить опыт, который удалось воплотить в ПО технологии, от опыта, который туда не вошел, но без котрого использовать технологию нельзя. Для пакетизации последнего нужны методические указания, учебные пособия, образцы, обучение и т.д. При внедрении технологии непакетизированный опыт требует дополнительной и, порой, очень объемной работы.

Технология UniTesK успешно прошла фазы 1 и 2 и в настоящий момент переходит к фазе 3.

Технология UniTesK [2] объединяет набор продуктов для тестирования промышленного ПО, реализованных для различных платформ — операционных систем, языков и сред программирования. UniTesK позволяет создать комплексную среду тестирования проекта по разработке и сопровождению ПО: тесты, средства взаимодействия тестовой платформы с тестируемой системой, методики тестирования (в режиме регулярного

автоматического регрессионного тестирования, различные варианты "ручного" тестирования и т.д.).

При тестировании с помощью UniTesK система рассматривается как "черный ящик", т.е. тестируется ее интерфейс — набор классов или процедур. Сами тесты генерируются автоматически по спецификации системы и тестовым сценариям. Спецификация описывает требования, предъявляемые к системе. Тестовый сценарий — это порядок обращения при тестировании к элементам интерфейса системы.

Технология UniTesK основана на идее автоматической генерации и исполнении тестов по тестовой спецификации системы и тестовым сценариям. Спецификация описывает ожидаемое поведение системы, создается на основе функциональных требований к системе, проектной документации и т.д. и является набором пред- и постусловий, а также инвариантов к элементам интерфейса системы. Тестовый сценарий описывает конечный автомат, определяющий тестируемые состояния системы и воздействия, осуществляемые на нее, в процессе исполнения теста.

Первые технологические решения группы RedVerst использовали в качестве языка спецификаций специализированный формальный язык RSL [9]. Однако потом было установлено, что для большинства разработчиков использование языков, основанных на математической логике и теории множеств [7], представляет серьезные сложности. И тогда для создания тестов перешли к использованию стандартных языков программирования (C, C++, Java), расширенных небольшим набором дополнительных конструкций. В данном проекте технология UniTesK рассматривалась на примере технологического решения CTesK-Lite, использующего в качестве языка спецификации C и интегрированного в Microsoft Visual Studio.

Для апробации были выбраны два проекта разработки встроенных систем: цифровой обработки звука и реализации IPv6-протокола.

Система цифровой обработки звука. Цифровая обработка аналогового сигнала предполагает преобразование входного сигнала в цифровую форму, обработку полученной цифровой последовательности на DSP-процессоре и обратное ее преобразование в аналоговую форму. Обработка цифровых последова-

тельностей на DSP-процессоре обычно заключается в выполнении некоторого набора численных алгоритмов, таких, как свертки, фильтры, определение спектра сигнала с помощью преобразования Фурье и т.д.

Система, рассматриваемая в данном проекте, реализована на процессоре DSP56362 фирмы Motorola [3]. Программная часть данной системы представляла собой набор преобразований над звуком (или эффектов), являющихся наборами процедур. Тестировалось около 35

Программный код данного проекта был написан на языках C и ассемблере. Все алгоритмы создавались сначала на языке C, потом транслировались в ассемблер целевого процессора и там дорабатывались.

Реализация IPv6-протокола. Internet Protocol version 6 (IPv6) [4] является новой версией Интернет-протокола, учитывающей современные требования к Интернет — потребность в расширении адресного пространства и более гибкое управление подсетями.

Система реализовывалась на встроенном устройстве, имевшем только несколько сетевых интерфейсов (Ethernet, ATM, ADSL) и последовательный порт для конфигурирования. Проект реализовывался на языке C и работал под управлением операционной системы реального времени. Тестировалась часть IPv6-протокола под названием IPv6 Core [4].

2 Описание апробации

Апробация выполнялась по схеме, представленой на рисунке 1.

Освоение продукта CTesK-Lite прошло быстро и без проблем. Продукт оказался компактным и надежным. При апробации использовался пилотный вариант CTesk-Lite, в результате чего было найдено незначительное количество ошибок, которые, однако, не помешали его использованию.

Критерии тестирования. Для проекта цифровой обработки звука были выбраны критерии двух типов:

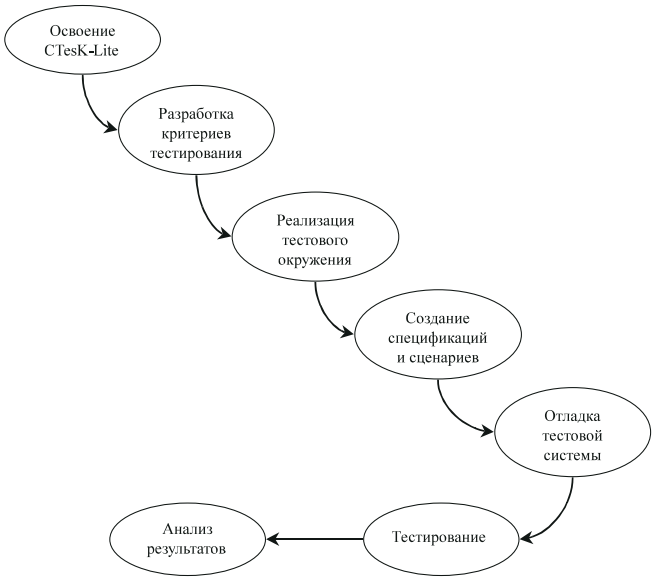

Рис. 1: Схема апробации

- сравнение результатов выполнения C-варианта системы на персональном компьютере под управленеим Windows и целевого кода системы на симуляторе;
- проверка качества реализации звуковых эффектов.

Критерии качества были выбраны следующие:

- система должна преобразовывать звуковые данные, задающие непрерывную кривую, в данные, задающие также непрерывную кривую;
- система должна генерировать выходные значения, которые находятся в диапазоне допустимых значений преобразователя цифрового представления сигнала в аналоговую форму;
- для любой входной звуковой кривой система не должна порождать некомфортный для человеческого уха шум.

Применение этого подхода было осложнено слабой математической подготовленностью тестеров в данной области.

Для IPv6-проекта тестовым критерием было соответствие системы IETF[1]-стандарту. Таким образом, фаза создания критериев тестирования практически отсутствовала в экспериментах с этим проектом.

Реализация тестового окружения. Сложность реализации тестового окружения для проекта цифровой обработки звука заключалась в необходимости создания эффективного медиатора — программной подсистемы, отвечающей за взаимодействие между системой и тестовым окружением. В проекте цифровой обработки звука тестируемая система работала на симуляторе процессора DSP56362, и в задачу медиатора входила корректная инициализация системы, управление ее работой, передача параметров тестируемым модулям и получение от них результатов. К сожалению, пакетный интерфейс этого симулятора не предоставлял возможность ставить точки останова, и их пришлось эмулировать в тестах. Это существенно замедлило скорость работы тестовой системы.

Тестовое окружение для IPv6-проекта было значительно проще, так как использовался "естественный" интерфейс тестируемой системы для посылки и получения сетевых пакетов. Медиаторы в данном случае представляли собой простую надстройку над интерфейсом портов (sockets).

Спецификации и сценарии, отладка тестовой системы. В соответствии с выработанными критериями были созданы спецификации и тестовые сценарии.

Отладка тестовой системы для IPv6-проекта проводилась на стандартной реализации этого протокола для операционных систем Linux и FreeBSD. Полученный таким образом тестовый набор стал самостоятельным программным продуктом. Он может быть повторно применен без каких-либо изменений к большому количеству реализаций (как встроенных, так и работающих на PC).

Для проекта по обработке звука отладка медиатора производилась созданием пробной тестовой спецификации, которая делала в точности то же, что и тестируемая подсистема. Отладка тестовой спецификации отдельно почти не производилась: было найдено лишь несколько случаев зацикливания тестов. Отлад-

ка тестов могла бы выражаться в изучении ситуаций, когда они что-то находили. Но такого не случалось, в частности, из-за медленной работы тестовой системы, не позволявшей пропускать большой объем тестовых данных.

Тестирование. Для проекта обработки звука была налажена система ночного тестирования, поскольку время обработки одного эффекта над пакетом входных данных занимало от 7 до 10 ч. Для IPv6-проекта тестирование проводилось вручную, поскольку итоговый тестовый пакет работал очень быстро — около 10 мин. и, кроме того, объем тестируемого кода, а также величина входных данных для тестов были существенно меньше, чем для проекта по цифровой обработке звука (см. таблицу).

3 Результаты

В таблице ниже приведены результаты апробации. Видно, что при составлении спецификации для проекта по обработке звука активно интервьюировались разработчики и анализировался программный код системы. Для IPv6-проекта использовалась только стандартная функциональная спецификация. При этом во втором случае труда на создание тестовой системы было потрачено существенно меньше, а результаты оказались лучше.

Отметим также, что размер тестовой системы в случае IPv6-проекта близок к размеру исходной системы, а в случае проекта обработки звука он составляет лишь четверть. Это объясняется тем, что для первого случая тестовая спецификация практически полностью реализует соответствующую часть системы. Отличие спецификации от реализации заключается в отсутствии ограничений на эффективность и размер используемой памяти для спецификации, что позволило применить более простые алгоритмы и структуры данных при ее разработке, без учета ограничений реального времени. Во втором случае тестовый код являлся принципиально иным, имел много переиспользуемых частей (в частности, подготовка исходных данных) и был проще, чем итоговая реализация системы.

Затраченные усилия и результаты	**Система обработки звука**	**Реализация стека IPv6**
Объем исходных текстов, проанализированных для создания тестовой спецификации, Кб.	48	0
Объем документации (включая стандарты), проанализированной для создания тестовой спецификации, Кб.	0	86
Общение с разработчиками для создания тестовой спецификации, дн.	7	0
Размер тестовой спецификаций, Кб.	20	20
Размер тестовых сценариев, Кб.	5	30
Размер медиаторов, Кб.	15	1
Количество найденных ошибок	2	3
Общий объем затраченных ресурсов на настойку технологии на тестируемую систему, чел/мес.	2	0.5
Общий объем созданного кода, Кб.	40	51
Отношение размера тестового набора (спецификация + тестовые сценарии+медиаторы) к размеру тестируемого кода системы	0.25	0.7

Наличие стандартных реализаций IPv6-протокола позволило существенно упростить отладку тестового окружения. Использование при тестировании “естестенного” интерфейса дало возможность создать очень простой медиатор.

Трудоемкость создания тестового окружения для проекта по цифровой обработке звука была связана с тем, что:

- проект являлся пилотным — для него не было строгих требований;

- тестеровщики не обладали нужной математической подготовкой в области цифровой обработки звука.

В проекте по цифровой обработке звука все ошибки были найдены только при изучении кода системы. В IPv6-проекте ошибки были найдены при эксплуатации тестового окружения, построенного на основе CTesk-Lite.

4 Анализ результатов

Проведенная апробация позволяет говорить о следующих свойствах технологии UniTesK-Lite с точки зрения ее "пакетизации":

1. ПО CTesK-Lite надежно и просто в использовании, его легко установить;

2. при создании тестового окружения удобно использовать общепринятый язык программирования, совпадающий с языком реализации системы, а также работать в знакомой среде программирования (в данном случае Microsoft Visual C++);

3. методика использования продукта CTesK-Lite, несмотря на наличие математических абстракций, была быстро освоена группой по апробации. Однако нужно отметить, что все члены группы имели классическое математическое образование. Есть опасения, что математические абстракции UniTesK (конечные автоматы, факторизация конечного автомата, пред/постусловия и т.д.) будут трудно усваиваться обычными разработчиками промышленного ПО. Тот факт, что в технологии вместо языков спецификаций, основанных на математической логике, используются широко распространенные языки программирования, является лишь первым шагом по инкапсуляции математического аппарата;

4. технология требует очень больших начальных ресурсов по ее настройке на конкретный проект. Сюда входит а) обучение методологии, содержащей математические абстракции б) изучение архитектуры продукта в) создание критериев тестирования для данного проекта. Продукт крайне трудно "посмотреть" в режиме простого ознакомления, т.е. без специально выделенных ресурсов;

5. в тех случаях, когда критерии тестирования очевидны, использование технологии UniTesK приносит результаты быстро и с небольшими затратами. Таким являлся Ipv6-проект (см. таблицу). Это возможно для систем, требования к которым стандартизированы, а также для систем, при разработке которых создана спецификация требований, размещенных на ПО (requirements allocated on the software) [6]. Как показывает опыт, последнее на практике бывает далеко не всегда — очень часто формализация требований остается на уровне бизнес-логики системы. Создать на их основе критерии автоматического тестирования очень не просто. С последним случаем мы столкнулись в проекте по обработке звука, где нам не удалось выразить формально и содержательно требования к системе, по которым были бы найдены ошибки;

6. для встроенных систем реального времени отдельной проблемой является реализация возможности взаимодействия тестового окружения и системы. В нашем случае стандартный симулятор процессора DSP56362 не предоставил пакетный интерфейс, поддерживающий приемлемое быстродействие тестового окружения. Поиск удовлетворительных решений вместе с созданием критериев тестирования является очень медленным процессом;

7. технология UniTesK имеет мощные средства для создания переиспользуемого тестового окружения. Это и модульная архитектура программных средств, и разделение тестовой спецификации, сценариев, медиаторов. Данное свойство UniTesK можно использовать как при создании решений для отдельных классов задач, так и для реализации общих активов в рамках семейств программных продуктов (product lines) [5].

Выводы

Таким образом, для успешного применения технологии UniTesK необходимо следующее:

- зрелый процесс разработки ПО (не ниже CMM II уровня);

- развитое управление требованиями (или готовность развивать эту область процесса);
- директивный управленческий стиль в компании [8] (например, военные или федеральные огранизации), со строгой вертикальной иерархией и высоким уровнем переиспользования активов процесса;
- наличие специалистов с математическим образованием для работы в тестировании;
- готовность организации тратить значительные ресурсы на наладку технологии;
- высокие требования к качеству создаваемого ПО;
- внедрение технологии или в масштабный проект и/или с перспективой распространения в другие проекты организации.

Встроенные системы являются перспективным направлением для технологии UniTesK. Однако для тиражирования UniTesK в этой предметной области необходимы дополнительные эксперименты, а также создание специальных методик, шаблонов и пр., в частности, необходимы хорошо разработанные методики построения медиаторов, тестовые критерии и т.д. Эта область перспективна для UniTesK, так как процесс разработки таких систем, как правило, является зрелым, там развито управление требованиями, цена ошибки в таких системах очень высока и поэтому есть готовность компаний тратить значительные средства на тестирование. Из-за сложности этого класса разработок в этой области активно создаются семейства программных продуктов.

Технология UniTesK снабжает пользователей архитектурой процесса тестирования, которую они могут по-разному реализовывать, а также развивать в следующих направлениях: а) процесс создания тестовых спецификаций может происходить в рамках анализа и проектирования системы, тем самым дополнительно упорядочиваяя эти фазы; б) тестовая система может быть интегрирована с базой данный дефектов, со средствами версионного контроля, автоматическими сборщиками проектов, генераторами отчетов и т.д. [1].

В заключение отметим, что технология UniTesK не является "легкой" в использовании. Фактически, она неявно предполагает, что ее пользователи должны стать столь же опытными в тестировании, как и разработчики технологии. То есть пользователи должны освоить опыт разработчиков, а не использовать его через простой внешний интерфейс.

Список литературы

[1] Оносовский В.В., Терехов А.Н. Организация работ в проекте RescueWare // Автоматизированный реинжиниринг программ. — СПб.: Изд-во СПбГУ, 2000. — С. 43-63.

[2] Bourdonov I.B., Kossatchev A.S., Kuliamin V.V., Petrenko A.K. UniTesK Test Suite Architecture // LNCS. — 2002. — Vol. 2391. — P. 77.

[3] DSP56300 Family Manual, revision 2.0. — Motorola, 1999. — 46 p.

[4] Hinden R., Deering S. Internet Protocol, Version 6 (IPv6) Specification. — IETF, Network Working Group, RFC 2460, 1998.

[5] Northrop L.M.. SEI's Software Product Line Tenets // IEEE Software. — 2000. — P. 32-41.

[6] Paulk M.C., Weber C.V., Garcia S.M., Chrissis M.B., Bush M. Key Practices of the Capability Maturity Model SM, Version 1.1: Technical Report CMU/SEI-93-TR-025 ESC-TR-93-178. — 1993. — 479 p.

[7] Petrenko A.K. Specification Based Testing: Toward Practice // LNCS. — 2001. — Vol. 2244. — P. 287-301.

[8] Romanovsky K. Generation-Based Software Product Line Evolution: A Case Study // Proc. of 2nd International Workshop "New Models of Business: Managerial Aspects and Enabling Technology". — 2002 — P. 178-186.

[9] The RAISE Specification Language. — Prentice Hall: 1992. — 397 p.