Опыт использования UniTESK как зеркало развития технологий тестирования на основе моделей

Иванников В. П., Петренко А. К., Кулямин В. В., Максимов А. В.
Институт системного программирования РАН (ИСП РАН), Москва
{ivan,petrenko,kuliamin,andrew}@ispras.ru

Аннотация. UniTESK — технология тестирования, основанная на формальных моделях (или спецификациях) требований к поведения программных или аппаратных компонентов. В статье описываются наиболее значимые применения технологии UniTESK в промышленных проектах, суммируется их опыт и оцениваются перспективные направления развития компонентных технологий тестирования на основе моделей.

Ключевые слова: тестирование на основе моделей; спецификации; верификация; автоматизированная генерация тестов

1. Введение

Тестирование на основе моделей (Model Based Testing, MBT) является одной из наиболее интенсивно развивающихся областей программной инженерии. Одна из причин такого активного развития — тот факт, что MBT находится на пересечении нескольких более широких областей. Эти области включают, в частности, методы описания, формализации и моделирования требований, методы анализа формальных моделей, методы статического и динамического анализа кода, методы управления уровнем абстракции и трансформации моделей. Такое положение вещей дает возможность технологиям MBT быстро адаптироваться к последним достижениям в смежных областях. Однако, до сих пор не существует готовых для промышленного использования инструментов MBT, которые можно было бы применять с достаточной эффективностью в широком классе проектов разработки и тестирования. Для дальнейшего развития технологий MBT необходим анализ причин такой ситуации, основанный на опыте их использования в последние 15-20 лет. Это поможет выявить проблемы существующих технологий и увидеть возможные пути их решения. В данной статье мы кратко описываем этапы развития технологии UniTESK (UNIfied TEsting and Specification toolKit,

унифицированный набор инструментов спецификации и тестирования) как одной из первых технологий, нацеленных на применение для тестирования широкого класса систем. По ходу статьи мы выделим позитивные и негативные итоги использования этой технологии. Эта статья является расширенным вариантом работы [36] и представляет собой анализ опыта промышленного использования технологий и инструментов, и поэтому не содержит постановок научных проблем или изложения новых техник и методов решения каких-либо задач. Тем не менее, проведенный анализ может быть полезен исследователям в области MBT.

Технология UniTESK [1,2] была создана на основе опыта, полученного при разработке системы автоматизированного тестирования KVEST [3] для ядра операционной системы реального времени. Работа над KVEST началась в 1994 году, когда термин «тестирование на основе моделей» еще не использовался. Этот термин появился в самом начале XXI столетия. Сейчас MBT очень быстро развивается. Технологии MBT поддерживаются многими энтузиастами и само понятие «тестирования на основе моделей» интерпретируется по-разному. Для аккуратного анализа места UniTESK среди других подходов нужно сначала определить, что подразумевается под этим термином в рамках технологии UniTESK.

Обычно говорят, что тестирование на основе моделей использует различные модели для построения тестов. Однако, у подобного утверждения очень много различных толкований — большинство исследователей и инженеров, использующих MBT на практике, подразумевают под этим термином более специфические техники тестирования. Первый водораздел — что именно является моделируемым объектом: некоторые моделируют поведение тестируемой системы (system under test, SUT), другие — окружение тестируемой системы, в частности, может моделироваться поведение тестов или тестовой системы как части этого окружения. В UniTESK моделируется поведение тестируемой системы. В рамках такого подхода выделяются методы MBT, использующие различные способы описания поведения. О первом из этих способов Ян Пелеска (Jan Peleska) говорит [4]: «поведение SUT задается моделью, которая представлена в том же стиле, как и модели, используемые для разработки». Такие спецификации или модели называются *исполнимыми*. Роль исполнимой модели может играть прототип реализуемой системы или другая модель, для которой есть понятие выполнения, например, конечный автомат, сеть Петри, машина с абстрактным состоянием (ASM) [5]. Примерами моделей другого вида, *неисполнимых*, служат алгебраические спецификации, а также программные контракты в виде пред- и постусловий функций. Различные виды моделей имеют свои достоинства и недостатки, более или менее подходят для тестирования систем различных типов. Так, при генерации тестов нужно строить не только тестовые данные и последовательности вызовов различных операций, но и другие артефакты, например, тестовые оракулы — компоненты тестов, автоматически оценивающие результаты работы SUT и выносящие решение, соответствуют

ли они требованиям или нет. Исполнимые модели не всегда подходят для генерации оракулов, но достаточно эффективно используются для построения тестовых последовательностей. Программные контракты упрощают генерацию оракулов, но для построения тестовых последовательностей не так удобны. Другими словами, для эффективного построения всех элементов тестовой системы лучше использовать модели различных типов. Возвращаясь к UniTESK, отметим, что основной вид моделей, используемый в этой технологии — это программные контракты. Однако, для генерации нетривиальных тестовых последовательностей в ней используются автоматные модели.

Технология UniTESK может быть реализована в рамках различных систем разработки и на различных языках. Сейчас используются реализации подхода UniTESK на основе языков C, C++, Java, Python, это, соответственно, инструменты CTESK, C++TESK, JavaTESK и PyTESK [6].

UniTESK является исследовательской разработкой ИСП РАН. Инструменты UniTESK доступны под свободной лицензией. Вместе с тем, имеется опыт промышленного использования этой технологии. Некоторые тестовые наборы, созданные с помощью UniTESK, используются для сертификации и аттестации промышленного программного обеспечения (ПО). Примером является тестовый набор OLVER [7] — один из самых больших тестовых наборов, созданных при помощи технологий MBT, уступающий лишь системе тестов, разработанной в рамках Инициативы по поддержке взаимодействия Microsoft (Microsoft Interoperability Initiative) [8].

2. Обзор применений UniTESK

Рассмотрим наиболее интересные примеры использования технологии UniTESK и полученные в рамках этого использования уроки.

Первым применением UniTESK был поддержанный Microsoft Research проект по разработке тестового набора для реализации IPv6 [9]. Проект стартовал в 2000 году. Тогда UniTESK только начинал создаваться, поэтому пришлось использовать облегченную реализацию технологии на языке C — CTESK-light. Несмотря на нестабильность инструмента удалось построить эффективный тестовый набор, который нашел дефекты, ранее не обнаруженные другими тестовыми наборами. Это был первый опыт использования контрактных спецификаций для тестирования телекоммуникационных протоколов. Было показано, что контрактные спецификации в сочетании с техникой тестирования систем с асинхронными интерфейсами, разработанной в рамках UniTESK [10,11] позволяют строить эффективные тесты. Эти тесты выявили больше ошибок и требовали меньше усилий для создания и сопровождения при заданном уровне качества тестирования, чем тесты, построенные по традиционным технологиям. Вместе с тем, опыт тестирования протоколов показал, что для эффективной генерации

последовательностей тестовых воздействий при тестировании протоколов полезно иметь и исполнимые модели.

Одним из первых опытов использования UniTESK для тестирования программных компонентов Java [12] через программный интерфейс (Application Programming Interface, API) стал проект тестирования одной из реализаций стандартной библиотеки поддержки времени исполнения Java. Разработка моделей и тестов не вызывала особых проблем, так как интерфейсы были хорошо документированы. Помимо интерфейсов на Java в системе также были интерфейсы на C++, но больших проблем и это не вызвало, поскольку архитектура тестов UniTESK предусматривает слой тестовых адаптеров. Более серьезные проблемы появились, когда началось собственно тестирование, в рамках которого генератор тестовой последовательности должен был работать на базе самой тестируемой системы, еще не стабильной в это время.

Одним из значимых примеров применения UniTESK на платформе Java является проект тестирования инфраструктуры распределенной информационной системы одного из крупных операторов мобильной связи, который продолжается и сейчас. Возможность формальной и строгой фиксации интерфейсов компонентов этой чрезвычайно большой и разнородной системы стала для заказчика самым главным преимуществом UniTESK по сравнению с другими инструментами функционального тестирования. В рамках проекта были формально специфицированы и протестированы сотни компонентов. Уже к концу первого года применения технологии положительный эффект проявился в ускорении сроков интеграции новых версий распределенной системы. Вместе с тем, вскрылась серьезная проблема. Если в предыдущих проектах применения UniTESK требования к большей части интерфейсов определялись стандартами или другими тщательно разработанными документами, здесь уровень документирования часто оказывался недостаточным для построения консистентных спецификаций. Восстановление документации или требований к интерфейсам в системах такого размера оказывается практически неразрешимой задачей, что не позволяет использовать MBT в полном объеме. О путях решения этой проблемы будет кратко сказано в Заключении.

Самым крупным примером применения UniTESK стал проект OLVER (Open Linux VERification) [7], который проводился в 2005-2007 годах при поддержке Министерства образования и науки РФ. Целью проекта была формальная спецификация интерфейсов стандарта Linux Standard Base (LSB), точнее его центральная части LSB Core. В LSB Core входят наиболее важные библиотеки операционной системы Linux, которые в значительной части реализуют стандарт POSIX. Строгое описание стандарта LSB и наличие набора тестов, который мог бы качественно проверить соответствие разных реализаций библиотек Linux требованиям стандарта, являются необходимыми условиями для обеспечения переносимости приложений для Linux с одного дистрибутива на другой. Проблема переносимости приложений под Linux является крайне

острой, поскольку сейчас доступно уже несколько сотен различных дистрибутивов. Результаты проекта опубликованы [7]. Были построены контрактные спецификации более чем 1500 интерфейсов на языке C. Инструментом моделирования и генерации тестов был выбран CTESK. В ходе проекта были выявлены проблемы в самих стандартах: LSB (ISO/IEC 23360) и The Single UNIX Specification, основную часть которого составляет стандарт POSIX.1 (он же IEEE Std 1003.1, он же ISO/IEC 9945, он же The Open Group Base Specifications Issue 6). Разработанный тестовый набор включен в пакет сертификационных тестов международного консорциума The Linux Foundation [13].

Опыт формализации интерфейсов большого индустриального стандарта и разработки тестового набора для него дал много полезных уроков. Одним из них стало понимание важности организации информационного и методического обеспечения такого проекта. Массив документации и исходных текстов библиотек Linux, особенно с учетом многочисленности версий и вариантов, связанных с различными аппаратными платформами, является огромным. Кроме того, в разработку собственно стандарта и в разработку его реализаций вовлечены тысячи людей, распределенных по всему миру. Из этого следует, что организация «документооборота» является одной из важнейших составляющих проектов такого калибра. В организационно-методическом плане мы столкнулись с тем, что обучение новых сотрудников и контроль за качеством спецификаций и тестов требует много усилий и при этом быстро достичь необходимого уровня профессионализма невозможно. То есть, масштабируемость проектов по использованию MBT в плане расширения числа участников проекта, задействованных в самой работе по спецификации требований и отладке тестов, — это одна из самых сложных проблем, мешающая широкому внедрению MBT.

Одной из методических проблем является выбор уровня абстракции, на котором строится модель. Более абстрактные модели или разделение моделей на два-три слоя, отличающиеся уровнем абстракции, упрощают задачу повторного использования моделей и тестов, при этом общий размер системы возрастает, и ее сложность также растет. В долгосрочном плане выгоднее иметь многослойные модели, в краткосрочном — модели, близкие по уровню детальности к реализации, конечно, если реализация уже есть. Профессиональный опытный верификатор умеет находить баланс между абстрактным описанием поведения, например, файловой системы и особенностями, деталями интерфейса конкретной реализации файловой системы. В UniTESK имеется специальная поддержка разделения уровней абстракции. В частности, специфика конкретных интерфейсов может быть скрыта в слой адаптеров. Выбор баланса во многом определяется долгосрочными планами по использованию и развитию моделей и тестового набора. То есть, такого рода работа требует достаточно широкого кругозора, чего трудно требовать то обычных инженеров-тестировщиков.

Результаты проекта OLVER впоследствии были использованы в разработке тестового набора для российской операционной системы реального времени ОС 2000/3000 [14]. Эта система поддерживает две группы интерфейсов. Первая отвечает требованиям POSIX, вторая — требованиям ARINC 653, международного стандарта для критических встроенных систем. Выделение уровня адаптеров, разделяющего модельное и реализационное представление интерфейсов, заложенное в архитектуру UniTESK, существенно упростило повторное использование OLVER в данном проекте.

Одновременно с началом работ по OLVER были развернуты работы по применению UniTESK для имитационного тестирования моделей отдельных блоков микропроцессоров [15]. Полученные тесты использовались при разработке российских микропроцессоров с архитектурой MIPS и микропроцессоров с элементами VLIM/EPIC. Размер типовых блоков в таких микропроцессорах — несколько миллионов вентилей. Для целей спецификации и генерации тестов не потребовалось больших изменений в инструментах, за основу был взят CTESK. В техническом плане привязка CTESK к API на языке C не стала проблемой, так как большинство симуляторов, работающих с языками моделирования логики микропроцессоров (High Level Design languages, HLD), например, VHDL или Verilog, предоставляют удобный интерфейс для взаимодействия с программами на C. Несколько изменилась семантика предусловий в контрактных спецификациях, они стали описывать не столько разрешенную область входных данных, сколько условия возможности выполнения микрооперации на соответствующем такте, что, кстати, характерно и для семантики предусловий асинхронных событий при тестировании параллелизма. Так же, как и в случае моделирования протоколов, выявилась потребность в использовании наряду с постусловиями, явных моделей поведения тестируемого устройства.

Как и в проектах по верификации программных систем одной из главных проблем, мешающих внедрению MBT при верификации аппаратного обеспечения (как и большинства других методов верификации), является отсутствие четких и детальных описаний функциональных требований к компонентам. Вместе с тем, ситуация в разработке микропроцессоров несколько лучше, так как в ее ходе принято наряду с HLD моделями строить и системные или архитектурные модели, описывающие семантику набора инструкций. Элементы таких архитектурных моделей можно использовать для восполнения недостающих знаний о поведении некоторых блоков микропроцессоров [16]. Хорошей новостью оказалось достаточно простое решение задачи распараллеливания выполнения теста на кластерах. Типичные размеры конечного автомата, который генерируется при выполнении теста одного сложного блока микропроцессора — это несколько миллионов узлов и десятки миллионов переходов. Оказалось, что генерация теста с помощью обхода неизвестного конечного автомата хорошо распараллеливается на кластерах до 200 узлов, с накладными расходами всего лишь 10-15%. При

этом надо помнить, что при имитационном тестировании высокоуровневой модели устройства основное время уходит на работу симулятора HDL, т.е. такая масштабируемость связана с возможностью запустить на каждом узле отдельный симулятор и выполнять параллельно много действий в различных состояниях. Масштабируемость распараллеливания тестирования систем других типов, скорее всего, будет не такой высокой.

Важно отметить задачи верификации, которые не удалось свести к моделированию на основе контрактных спецификаций, из-за чего они дали толчок для разработки новых методов MBT. В первую очередь следует упомянуть задачу тестирования компиляторов и задачу тестирования микропроцессора в целом, так называемый «core testing». В обоих случаях это задачи системного тестирования, где требуется подавать на вход большого «черного ящика» тестовые воздействия (в нашем случае это тестовые программы, которые подаются на вход компилятору или загружаются в память симулятора микропроцессора), но при этом интересно тестировать не все подряд, а лишь некоторые заданные режимы или заданную группу модулей компилятора или процессора. В случае тестирования компиляторов был разработан инструмент OTK, который использовался для тестирования оптимизирующих компиляторов Intel и Simulink [17,18]. Он позволяет нацеливаться на заданные виды оптимизаций. Для системного тестирования микропроцессоров был разработан инструмент MicroTESK [19,20,21]. Основной задачей этого инструмента была проверка разнообразных ситуаций, связанных с наиболее сложными подсистемами управления памяти: блоком трансляции адресов, кэшами разных уровней или блоком управления памятью в целом.

3. Выводы и направления дальнейшего развития

Начнем с позитивных выводов.

3.1. Позитивные выводы по современному состоянию MBT

- История применений UniTESK в промышленных разработках подтверждает мировой опыт [22]: MBT можно эффективно применять в промышленных проектах, при этом по сравнению с традиционным тестированием этот подход дает уникальное преимущество — удается находить достаточно много ошибок в требованиях, которые часто существенно дороже ошибок в реализации.
- Удается достичь уровень тестового покрытия существенно выше традиционного (даже в сравнении с тестированием «белого ящика»). Так, в случае применения OTK для тестирования компилятора gcc удалось довести уровень покрытия до 95%, а в случае компилятора

Intel до 75%, что существенно превышало уровень покрытия? полученного традиционными тестами [18].

- Хотя многослойная структура спецификаций (несколько уровней абстракции) редко используется на практике, явное выделение слоя адаптеров упрощает перенос и сопровождение, и, наоборот, его отсутствие существенно усложняет разработку тестовых наборов, что было показано в программе Microsoft Interoperability Initiative [23].
- Генерация тестовых последовательностей в виде обхода неявно заданного конечного автомата хорошо распараллеливается и позволяет использовать вычислительные мощности кластеров при накладных расходах в 10-15%, по крайней мере в случае тестирования моделей микропроцессоров.
- Растет востребованность MBT в области критических систем и приложений. Это, в частности, находит отражение в стандартах, определяющих требования к процессам разработки таких систем, например, в DO178C [24] и в Common Criteria [25].

3.2. Негативные аспекты современного состояния дел в области MBT

- Самая главная трудность, препятствующая широкому распространению MBT — это отсутствие достаточно четких и детальных моделей (спецификаций). Такое положение дел часто является не только следствием недостаточного внимания к разработке спецификаций или экономии средств, часто основная причина его — отсутствие грамотных специалистов, которые одновременно владеют предметной областью и в состоянии построить модель, необходимую для построения тестов.
- Чаще всего в MBT используются модели, создаваемые специально для дальнейшего построения тестов на их основе. Однако из-за постепенного распространения разработки на основе моделей (Model-Driven Development, MDD) на руках разработчиков возникают другие модели. Создание особых моделей для тестирования при этом воспринимается как неприемлемые дополнительные расходы. Оказывается, что при тщательном планировании многие компоненты моделей разработки можно использовать и для генерации тестов, что показано, например, в диссертации М. М. Чупилко [16]. Однако получение реальной пользы от использования моделей разработки при тестирования требует дополнительной квалификации и тщательного анализа получаемых результатов — иначе чаще всего тесты, созданные по тем же лекалам, что и тестируемая система, просто не в состоянии обнаружить в ней никаких ошибок.

- Двуязычные системы генерации тестов типа первых версий UniTESK и SpecExplorer [26], а также специальные спецификационные нотации, даже приближенные к языкам программирования, например, JML [27], затрудняют внедрение технологий MBT. Двуязычные нотации требуют специальной подготовки персонала и затрудняют развитие созданных на их основе тестовых наборов. При этом современные ОО языки уже имеют развитые средства, которые позволяют описывать спецификации средствами базового языка [26-31].

3.3. Направления развития

- Для обеспечения высокой эффективности использования инструментов MBT при тестировании разнообразных систем нужно обеспечивать разнообразие поддерживаемых ими парадигм моделирования, в частности, контрактные спецификации совместно с, например, конечными автоматами или структурами Крипке. Каждый из видов моделей нацелен на облегчение анализа лишь определенных аспектов поведения системы и может затруднять анализ других. К сожалению, чаще всего наиболее выгодно использование разных моделей на разных уровнях абстракции, и все их приходится создавать вручную, без возможности автоматизации перехода от одной модели к другой. Любые продвижения в направлении автоматизации таких сложных, связанных с изменением уровня абстракции, трансформаций моделей, могли бы оказать значительную помощь в практическом использовании основанных на них методов MBT.
- Требуется развитие средств моделирования и описания спецификаций для целей MBT. Хотя наблюдается прогресс в области технологий разработки с использованием проблемно-ориентированных языков (Domain Specific Languages, DSL), в практическом плане системы, базирующиеся на универсальных языках выигрывают за счет наличия большого числа специалистов, владеющих такими языками, и широкой их поддержки разнообразными инструментами. То же можно сказать и про моноязычные системы — они выигрывают у мультиязычных за счет большей простоты использования и развития.
- Современные достижения в области статического и динамического анализа позволяют интегрировать эти техники в системы MBT, причем объектом анализа должны быть как модели, так и программный код тестируемой системы.
- Для преодоления проблем, связанных с катастрофическим недостатком детальных спецификаций в реальной практике, нужно развивать и внедрять инструменты работы с требованиями и моделями (см., например, [33]), в частности, с моделями программно-

аппаратных систем [34,35]. Для многокомпонентных систем нужно интегрировать MBT со средствами выявления и реинжиниринга архитектурных и проектных решений.

Литература

[1]. Bourdonov, I.B., Kossatchev, A.S., Kuliamin, V.V., Petrenko, A.K.: UniTesK Test Suite Architecture. In: FME 2002. LNCS, vol. 2391, pp. 77-88. Springer-Verlag (2002)

[2]. Кулямин, В.В., Петренко, А.К., Косачев, А.С., Бурдонов, И.Б.: Подход UniTesK к разработке тестов. Программирование, 29(6), 25-43 (2003)

[3]. Bourdonov, I.B, Kossatchev, A.S., Petrenko, A.K., Galter. D.: KVEST: Automated Generation of Test Suites from Formal Specifications. In: Proceedings of Formal Method Congress, Toulouse, France, 1999. LNCS, vol. 1708, pp. 608-621 (1999)

[4]. Peleska, J.: Industrial-Strength Model-Based Testing – State of the Art and Current Challenges. Invited Talk. In: Petrenko, A.K., Schlingloff, H. (eds.) Proceedings Eighth Workshop on Model-Based Testing (MBT 2013), Rome, Italy, 17th March 2013. Electronic Proceedings in Theoretical Computer Science, 111, pp. 3–28. DOI: 10.4204/EPTCS.111.1 (2013)

[5]. Börger, E., Stärk, R.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag (2003)

[6]. UniTESK technology, http://unitesk.ispras.ru

[7]. OLVER project, http://linuxtesting.org

[8]. Microsoft Interoperability Initiative, http://www.microsoft.com/openspecifications

[9]. Пакулин, Н.В., Хорошилов, А.В.: Разработка формальных моделей и тестирование соответствия для систем с асинхронными интерфейсами и телекоммуникационных протоколов. Программирование, 33 (6), 26-55 (2007)

[10]. Хорошилов, А.В.: Спецификация и тестирование компонентов с асинхронными интерфесами. Диссертация на степень к.ф.-м.н., Москва (2006)

[11]. Kuliamin, V.V., Petrenko, A.K., Pakulin, N.V.: Extended Design-by-Contract Approach to Specification and Conformance Testing of Distributed Software. In: Proceedings of WMSCI'2005, Orlando, USA, July 10-13, 2005. Model Based Development and Testing, v. VII, pp. 65-70 (2005)

[12]. Bourdonov, I.B., Demakov A.V., Jarov, A.A., Kossatchev, A.S., Kuliamin, V.V., Petrenko, A.K., Zelenov, S.V.: Java Specification Extension for Automated Test Development. In: Proceedings of PSI'01. LNCS, vol. 2244, pp. 301-307. Springer-Verlag (2001)

[13]. The Linux Foundation consortium. LSB certification test suite, http://ispras.linuxbase.org/index.php/LSB_Certification_System

[14]. Maksimov, A.V.: Requirements-based conformance testing of ARINC 653 real-time operating systems. In: Proceedings of the Data Systems In Aerospace (DASIA 2010) conference, 2010. ESA SP-682, ISBN 978-92-9221-246-9 (2010)

[15]. Иванников, В.П., Камкин, А.С., Косачев, А.С., Кулямин, В.В., Петренко, А.К.: Использование контрактных спецификаций для представления требований и функционального тестирования моделей аппаратуры. Программирование, 33(5), 47-61 (2007).

[16]. Chupilko, M.M.: Developing Test Systems of Multi-Modules Hardware Designs. ISSN 0361-7688, Programming and Computer Software, 2012, Vol. 38, No. 1, pp. 34-42. Pleiades Publishing, Ltd. (2012)

[17]. Zelenov, S.V., Zelenova, S.A.: Model-Based Testing of Optimizing Compilers. In: Proc. of the 19th IFIP TC6/WG6.1 International Conference on Testing of Software and Communicating Systems – 7th International Workshop on Formal Approaches to Testing of Software (TestCom/FATES 2007). LNCS, vol. 4581, pp. 365-377. Springer-Verlag, Berlin (2007)
[18]. Zelenov, S.V., Silakov, D.V., Petrenko, A.K., Conrad, M., Fey I.: Automatic Test Generation for Model-Based code Generators. In: IEEE ISoLA 2006 Second Intern. Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Paphos, Cyprus, pp. 68-75 (2006)
[19]. Камкин, А.С.: Метод автоматизации имитационного тестирования микропроцессоров конвейерной архитектуры на основе формальных спецификаций. Диссертация на степень к.ф.-м.н., Москва (2009)
[20]. Корныхин, Е.В.: Метод автоматизации генерации тестовых программ для верификации MMU. Диссертация на степень к.ф.-м.н., Москва (2010)
[21]. Kamkin, A.S., Tatarnikov, A.: MicroTESK: An ADL-Based Reconfigurable Test Program Generator for Microprocessors. In: Proceedings of the 6th Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2012), May 30-31, 2012, Perm, Russia (2012)
[22]. MBT survey, http://www.robertvbinder.com/docs/arts/MBT-User-Survey.pdf
[23]. Grieskamp, W.: Microsoft's Protocol Documentation Program: A Success Story for Model-Based Testing. In: Testing – Practice and Research Techniques. Lecture Notes in Computer Science, vol. 6303, p. 7 (2010)
[24]. Adams, C.: Safety-critical software for mission-critical applications to get boost with release of DO-178C. Military & Aerospace Electronics, 10 (2010)
[25]. Common Criteria, http://www.commoncriteriaportal.org
[26]. SpecExplorer, http://research.microsoft.com/en-us/projects/specexplorer
[27]. The Java Modelling Language (JML), http://www.eecs.ucf.edu/~leavens/JML/index.shtml
[28]. Pakulin, N.V.: Integrated Modular Avionics: New Challenges for MBT. In: ETSI TTCN-3 User Conference and Model Based Testing Workshop, Bangalore, India, 11-14 June 2012 (2012)
[29]. Code Contracts, http://research.microsoft.com/en-us/projects/contracts
[30]. C++TESK, http://forge.ispras.ru/projects/cpptesk-toolkit
[31]. Кулямин, В.В.: Компонентная архитектура среды для тестирования на основе моделей. программирование, 36(5), 54-75 (2010)
[32]. Kuliamin, V.V.: Multi-paradigm Models as Source for Automated Test Construction. In: Proceedings of the 1-st Workshop on Model Based Testing (MBT'2004, in ETAPS'2004), Barcelona, Spain, March 27-38, 2004, Electronic Notes in Theoretical Computer Science, 111:137-160, Elseveir, (2005)
[33]. ReQuality tool, http://requality.org/en/doc.en.html
[34]. Khoroshilov, A.V., Albitskiy, D., Koverninskiy, I.V., Olshanskiy, M.Yu., Petrenko, A.K., Ugnenko, A.A.: AADL-Based Toolset for IMA System Design and Integration. SAE Int. J. Aerosp. 5(2), DOI:10.4271/2012-01-2146 (2012)
[35]. Systems Modeling Language (SysML), http://www.sysml.org
[36]. Petrenko, A.K., Kuliamin, V.V., Maksimov A.V.: UniTESK: Component Model Based Testing. Proceedings of ICTERI 2013.

How the story of UniTESK technology applications mirrors development of model based testing

Ivannikov V.P., Petrenko A.K., Kuliamin V.V., Maksimov A.V.
ISP RAS, Moscow, Russia
{ivan,petrenko,kuliamin,andrew}@ispras.ru

Abstract. UniTESK (UNIfied TEsting and Specification toolKit) is a testing technology based on formal models (or specifications) of requirements to behavior of software or hardware components. It was created with experience gained during development of the framework for automated testing of a real-time operating system kernel during 1994-2000. Software contracts were the main kind of models in the first version of the technology. Automata models were also implemented to support generation of complicated test sequences. UniTESK was first used in 2000 in the development of the test suite for IPv6 implementation. It was the first experience of using contract specifications in testing of implementations of telecommunication protocols. This project demonstrated that contract specifications in combination with the technique of testing software systems with asynchronous interface developed within UniTESK are very effective. Applications of UniTESK to testing software components in Java include the project on testing of implementation of standard library for Java runtime support and the project on testing of infrastructure of information system for one of large mobile operators in Russia. The most significant application of UniTESK happened in 2005-2007 in the Open Linux Verification project. Formal specifications and tests for interfaces of the Linux Standard Base were created during the project. These results then were used in development of the test suite for Russian avionics real-time operating system.

Positive lessons learned during development and using UniTESK include effective application of model based testing methods in large industrial projects, high level of test coverage achieved by the generated tests, applicability of model based testing to critical systems and applications. Negative lessons include the lack of well-defined and detailed models and specifications, extra development expenses caused by creation of test specific models, problems with introduction of model based testing technologies using bilingual test generation systems and specific notations.

Keywords: model based testing, specification, verification, automated test generation

References

[1]. Bourdonov, I.B., Kossatchev, A.S., Kuliamin, V.V., Petrenko, A.K.: UniTesK Test Suite Architecture. In: FME 2002. LNCS, vol. 2391, pp. 77-88. Springer-Verlag (2002)
[2]. V. V. Kuliamin, A. K. Petrenko, A. S. Kossatchev, I. B. Burdonov: The UniTesK Approach to Designing Test Suites. Programming and Computer Software, 29(6), 310-322 (2003)
[3]. Bourdonov, I.B, Kossatchev, A.S., Petrenko, A.K., Galter. D.: KVEST: Automated Generation of Test Suites from Formal Specifications. In: Proceedings of Formal Method Congress, Toulouse, France, 1999. LNCS, vol. 1708, pp. 608-621 (1999)

[4]. Peleska, J.: Industrial-Strength Model-Based Testing – State of the Art and Current Challenges. Invited Talk. In: Petrenko, A.K., Schlingloff, H. (eds.) Proceedings Eighth Workshop on Model-Based Testing (MBT 2013), Rome, Italy, 17th March 2013. Electronic Proceedings in Theoretical Computer Science, 111, pp. 3–28. DOI: 10.4204/EPTCS.111.1 (2013)
[5]. Börger, E., Stärk, R.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag (2003)
[6]. UniTESK technology, http://unitesk.ispras.ru
[7]. OLVER project, http://linuxtesting.org
[8]. Microsoft Interoperability Initiative, http://www.microsoft.com/openspecifications
[9]. N. V. Pakulin, A. V. Khoroshilov: Development of formal models and conformance testing for systems with asynchronous interfaces and telecommunications protocols. Programming and Computer Software, 33 (6), 316-335 (2007)
[10]. A. V. Khoroshilov: Specifikacija i testirovanie komponentov s asinhronnymi interfesami [Specification and testing of components with asynchronous interfaces]. Dissertacija na stepen' k.f.-m.n. [PhD Thesis], Moskva (2006)
[11]. Kuliamin, V.V., Petrenko, A.K., Pakulin, N.V.: Extended Design-by-Contract Approach to Specification and Conformance Testing of Distributed Software. In: Proceedings of WMSCI'2005, Orlando, USA, July 10-13, 2005. Model Based Development and Testing, v. VII, pp. 65-70 (2005)
[12]. Bourdonov, I.B., Demakov A.V., Jarov, A.A., Kossatchev, A.S., Kuliamin, V.V., Petrenko, A.K., Zelenov, S.V.: Java Specification Extension for Automated Test Development. In: Proceedings of PSI'01. LNCS, vol. 2244, pp. 301-307. Springer-Verlag (2001)
[13]. The Linux Foundation consortium. LSB certification test suite, http://ispras.linuxbase.org/index.php/LSB_Certification_System
[14]. Maksimov, A.V.: Requirements-based conformance testing of ARINC 653 real-time operating systems. In: Proceedings of the Data Systems In Aerospace (DASIA 2010) conference, 2010. ESA SP-682, ISBN 978-92-9221-246-9 (2010)
[15]. V. P. Ivannikov, A. S. Kamkin, A. S. Kossatchev, V. V. Kuliamin, A. K. Petrenko: The use of contract specifications for representing requirements and for functional testing of hardware models. Programming and Computer Software, 33(5), 272-282 (2007).
[16]. Chupilko, M.M.: Developing Test Systems of Multi-Modules Hardware Designs. ISSN 0361-7688, Programming and Computer Software, 2012, Vol. 38, No. 1, pp. 34-42. Pleiades Publishing, Ltd. (2012)
[17]. Zelenov, S.V., Zelenova, S.A.: Model-Based Testing of Optimizing Compilers. In: Proc. of the 19th IFIP TC6/WG6.1 International Conference on Testing of Software and Communicating Systems – 7th International Workshop on Formal Approaches to Testing of Software (TestCom/FATES 2007). LNCS, vol. 4581, pp. 365-377. Springer-Verlag, Berlin (2007)
[18]. Zelenov, S.V., Silakov, D.V., Petrenko, A.K., Conrad, M., Fey I.: Automatic Test Generation for Model-Based code Generators. In: IEEE ISoLA 2006 Second Intern. Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Paphos, Cyprus, pp. 68-75 (2006)
[19]. A. S. Kamkin: Metod avtomatizacii imitacionnogo testirovanija mikroprocessorov konvejernoj arhitektury na osnove formal'nyh specifikacij [Method for automating simulation testing microprocessor pipeline architecture based on formal specifications]. Dissertacija na stepen' k.f.-m.n. [PhD Thesis], Moskva (2009)

[20]. E. V. Kornykhin: Metod avtomatizacii generacii testovyh programm dlja verifikacii MMU [Method for automating simulation testing method microprocessors automation test program generation for verification of MMU]. Dissertacija na stepen' k.f.-m.n. [PhD Thesis], Moskva (2010)
[21]. Kamkin, A.S., Tatarnikov, A.: MicroTESK: An ADL-Based Reconfigurable Test Program Generator for Microprocessors. In: Proceedings of the 6th Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2012), May 30-31, 2012, Perm, Russia (2012)
[22]. MBT survey, http://www.robertvbinder.com/docs/arts/MBT-User-Survey.pdf
[23]. Grieskamp, W.: Microsoft's Protocol Documentation Program: A Success Story for Model-Based Testing. In: Testing – Practice and Research Techniques. Lecture Notes in Computer Science, vol. 6303, p. 7 (2010)
[24]. Adams, C.: Safety-critical software for mission-critical applications to get boost with release of DO-178C. Military & Aerospace Electronics, 10 (2010)
[25]. Common Criteria, http://www.commoncriteriaportal.org
[26]. SpecExplorer, http://research.microsoft.com/en-us/projects/specexplorer
[27]. The Java Modelling Language (JML), http://www.eecs.ucf.edu/~leavens/JML/index.shtml
[28]. Pakulin, N.V.: Integrated Modular Avionics: New Challenges for MBT. In: ETSI TTCN-3 User Conference and Model Based Testing Workshop, Bangalore, India, 11-14 June 2012 (2012)
[29]. Code Contracts, http://research.microsoft.com/en-us/projects/contracts
[30]. C++TESK, http://forge.ispras.ru/projects/cpptesk-toolkit
[31]. V. V. Kuliamin: Component architecture of model-based testing environment. Programming and Computer Software, 36(5), 289-305 (2010)
[32]. Kuliamin, V.V.: Multi-paradigm Models as Source for Automated Test Construction. In: Proceedings of the 1-st Workshop on Model Based Testing (MBT'2004, in ETAPS'2004), Barcelona, Spain, March 27-38, 2004, Electronic Notes in Theoretical Computer Science, 111:137-160, Elseveir, (2005)
[33]. ReQuality tool, http://requality.org/en/doc.en.html
[34]. Khoroshilov, A.V., Albitskiy, D., Koverninskiy, I.V., Olshanskiy, M.Yu., Petrenko, A.K., Ugnenko, A.A.: AADL-Based Toolset for IMA System Design and Integration. SAE Int. J. Aerosp. 5(2), DOI:10.4271/2012-01-2146 (2012)
[35]. Systems Modeling Language (SysML), http://www.sysml.org
[36]. Petrenko, A.K., Kuliamin, V.V., Maksimov A.V.: UniTESK: Component Model Based Testing. Proceedings of ICTERI 2013.