XXXII Неделя науки СПбГПУ. Материалы межвузовской научно-технической конференции. Ч.V : С.10-11

УДК 681.3.06

Л.В.Дворянский (5 курс, ИУС), П.В.Дробинцев (асп., ИУС),
В.П.Котляров, к.т.н., проф.

ПЛАТФОРМЕННО-НЕЗАВИСИМЫЙ МЕТОД ТЕСТИРОВАНИЯ

С тех пор, как появилось понятие качества программного продукта (ПП) об актуальности тестирования говорить не приходится. Тестирование просто необходимо в жизненном цикле разработки программного обеспечения (ПО) организации. Любая автоматизация этого процесса несет в себе снижение себестоимости и повышение качества конечного продукта. Одна из проблем автоматизации тестирования - автоматический запуск тестовых наборов. Если разрабатываемый ПП является кросс-платформенным, возникает проблема доказательства качества работы ПП на всех целевых платформах. Из этого следует необходимость кроссплатформенности самих тестовых наборов. Эта проблема не кажется серьезной, если мы имеем дело с тестовым набором, содержащим небольшое количество тестов. В этом случае возможно написание отдельных тестовых наборов для каждой платформы. Однако в реальной жизни приходится иметь дело с сотнями и даже тысячами тестов. Проблема заключаетсядаже не только в том, что нужно написать в N раз больше строк кода (где N – число целевых платформ), но и в обеспечении идентичности тестовых наборов для каждой платформы. Если тест добавлен в тестовый набор для Windows и не добавлен в набор для Unix, то потом будет весьма затруднительно найти несоответствие. Анализу и решению этой проблемы посвящена настоящая работа.

Решением данных проблем было бы создание такой системы автоматического запуска тестового набора, которая бы одинаково работала на всех платформах. Соответственно, проблемы обеспечения идентичности и содержание большего объема тестового кода превратились бы в проблему создания такой системы. Проблемы кросс платформенности связанны с различием в бинарных форматах данных. Для решения этих трудностей используются скриптовые языки “data-driven” или управляемые данными приложения в совокупности с данными в открытом формате, доступном для любых платформ. Примером таких форматов являются простой ASCII-текст, base64, XML. А в качестве скриптового языка можно выбрать любой подручный язык, реализация которого есть на всех интересующих нас целевых платформах. Однако, возникает вопрос: что будет, если сама тестируемая система (ТС) является платформенно-зависимой (ПЗ) или использует внешние ПЗ части. Рассмотрим наиболее распространенные ситуации:

1) тестируемая система написана на одном из компилируемых языков, и после компиляции мы, по сути, имеет несколько утилит с одинаковым поведением, но каждая запускается только под свою платформу;

2) тестируемая система использует ПЗ бинарные форматы, такие как динамически подключаемые библиотеки;

3) в разных операционных системах приняты разные форматы файловых имен путей;

4) для тестирования системы необходима функциональность, которую предоставляют ПЗ утилиты (к примеру “rm” в Unix и “del” в Windows).

Для решения подобных проблем целесообразно выделить ПЗ поведение в отдельный ПЗ скрипт (ПЗС), который будет содержать набор специфичных процедур. В теле таких процедур осуществляется определение, какая именно платформа используется, и в зависимости от этого определяются дальнейшие действия и возвращаемый результат. Таким

образом, скрипт, вызывающий эти процедуры, будет вести себя по-разному, в зависимости от платформы на которой он выполняется, однако сам меняться не будет.

Таким образом, технология создания подобной системы следующая:

1) Планируется структура директорий для тестового набора. (Простейшая двухуровневая структуры - по одной папке для каждого теста, которые хранятся внутри одной папки для тестового набора.)

2) Создается ПЗС, в который выносятся все ПЗ части системы запуска тестов. (Рекомендуется организовывать их в виде процедур).

3) Внутри папок для тестов пишутся скрипты запуска тестов на выбранном языке.

4) Внутри папки для тестового набора создается описательный файл в открытом формате, содержащий данные о расположении тестов набора и правила проверки их результатов.

5) Внутри папки для тестового набора пишется скрипт запуска тестового набора, который для своей работы использует описательный файл.

К каждому скрипту запуска теста присоединяется ПЗС. Расположение ПЗС указывается как относительный путь или через переменную окружения. Так как ПЗС присоединен к каждому скрипту, выгодно все настройки окружения, такие как пути к ТС, вынести именно в ПЗС.

Такая методология была успешно применена для тестирования SIC (набора утилит автоматизации тестирования) с числом тестов чуть более 2200. В качестве скриптового языка был выбран TCL (Tool Command Language) как весьма гибкий язык, для которого имеются реализации под необходимые платформы. Первоначально тестирование осуществлялось только под платформой Windows. Однако после конвертации тестов под платформенно-независимый метод тестирования было выявлены дефекты в работе SIC под платформой Solaris, которые не проявились под Windows. Таким образом, примененная технология способствовала повышению качества SIC.