XXXI Неделя науки СПбГПУ. Материалы межвузовской научной конференции. Ч. VI: С.32-34, 2003.

УДК 658.512.011.56: 681.3.06

А.О. Некрасов (асп., каф. ИУС), Е.А. Савельева (6 курс, каф. ИУС),
А.В. Гаригин (5 курс, каф. ИУС), В.П. Котляров, к.т.н., проф.

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ GUI МНОГОПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ

Данная статья посвящена вопросам автоматизации тестирования приложений с графическими пользовательскими интерфейсами на платформах UNIX/X Window и MS Windows. Так же, данный подход можно использовать для автоматизации любых процессов, включающих взаимодействие с элементами графического интерфейса пользователя, включая удаленное администрирование.

Задача состоит в эмуляции пользовательских действий на консоли таким образом, чтобы не создавать для тестируемого приложения дополнительных побочных эффектов. Например, широко известное приложение QA Partner использует для контроля графических объектов специальные тестовые интерфейсы библиотеки MSC, что не позволяет утверждать, что приложение будет вести себя таким же образом и в реальной ситуации. Так же, использование специальных интерфейсов создает дополнительные ограничения на способ реализации самого тестируемого приложения, что тоже не всегда приемлемо.

Технически задача сводится к двум частям – эмуляция входных воздействий клавиатуры и мыши и проверка содержимого экрана в определенные моменты времени. Для каждой платформы существуют свои сложности и особенности, о которых пойдет речь ниже.

Выбор тестового языка.

Возникла необходимость выбора языка для написания тестов. При этом учитывались следующие требования к тестовому языку:

- Переносимость
- Расширяемость
- Наличие стандартных конструкций (if, for...)
- Модульность

При выборе рассматривались два варианта – написание своего языка и использование уже существующих.

Написание своего языка. Это привело бы к написанию достаточно большого количества функций (конструкций), реализация которых уже имеется в существующих языках, к созданию интерпретатора и т. д. В общем, пришлось бы создавать заново то, что уже написано и может быть с успехом переиспользовано. К тому же, некоторым нужным требованиям удовлетворяли уже существующие языки и надо было просто выбрать наиболее подходящий для данной задачи. Поэтому вариант написания своего языка не рассматривался далее.

Использование таких языков, как: C, Java, Tcl, Perl. Язык программирования C является слишком низкоуровневым. Написание конечных тестов на C оказалось бы достаточно трудоёмким и громоздким. При выборе Java как тестового языка возникли бы проблемы с расширяемостью - было бы достаточно сложно написать новые команды.

При примерно одинаковых возможностях Perl и Tcl был выбран Tcl.

Tcl обладает всеми перечисленными свойствами. Этот язык можно интегрировать в разные операционные системы и среды программирования. Тем самым выполняется условие переносимости тестового языка. Он включает в себя достаточное для реализации поставленной задачи количество стандартных конструкций. Всё это позволяет строить на его основе скрипты, достаточно простые и удобные для написания тестов (по сравнению с C и Java). Есть возможность дополнения языка пользовательскими библиотеками, содержащими новые команды. Каждая команда имеет ассоциированную с ней командную процедуру, которая и содержит всю логику выполнения команды. Для встроенных в интерпретатор команд эти

процедуры написаны на C и соответствующие команды доступны сразу после запуска интерпретатора. Для команд, определяемых пользователем это либо написанные на C процедуры, хранящиеся в динамически загружаемых библиотеках, либо Tcl скрипты выполняемые в интерпретаторе и создающие новые команды. Соответственно, существует возможность расширения данного языка.

Особенности реализации на X. Система X Window является сетевой системой управления окнами. Дисплейный сервер X работает на компьютерах, как с монохромными, так и с цветными дисплеями. Сервер распределяет ввод пользователя и принимает запросы вывода от различных клиентских программ, которые могут быть расположены как на той же машине, так и на любой машине в сети. Xlib это библиотека подпрограмм на языке C, которую прикладные программы (клиенты) используют как интерфейс с оконной системой, использующей для взаимодействия потоковое соединение.

Эмуляция нажатия на кнопки мыши и клавиатуры: для реализации данных функций используется тестовое расширение X - XTestExtension.

Расширение X - это фрагмент программного кода, который расширяет функциональность сервера X и добавляет в него новые возможности, отсутствующие в базовом протоколе X, например, непосредственную поддержку трехмерной графики, улучшенную обработку графических данных, возможность размещения изображений в общей памяти и т. д. Каждое расширение модифицирует сервер X и организует в нем поддержку расширенного протокола.

Работа с окнами в графических приложениях: Snapshots – сохранение изображения окна в файле и последующий вывод его этого изображения на экран.

Для реализации данной функции, а также для возможности последующего сравнения нескольких snapshots и их маскирования можно было использовать несколько путей. Сначала предполагалось побайтовое считывание данных из структуры XImage, содержащей параметры изображения и соответствующие данные. Каждый пиксел изображения кодируется соответствующим количеством битов (в зависимости от количества цветовых плоскостей, типа дисплея и т. д.). Но точное расположение битов, кодирующих каждый пиксел в поле структуры, неизвестно. Поэтому для работы со snapshots используются стандартные функции XGetPixel и XPutPixel для сохранения значения пиксела и вывода его на экран, соответственно. Значения пикселов сохраняются в файле, а затем считываются оттуда для восстановления изображения на экране.

Стабилизация теста: для стабильности теста необходим Flush. Он используется для последовательного своевременного выполнения команд. Большинство функций Xlib производят добавление запросов в буфер вывода. Позже эти запросы асинхронно выполняются X сервером. Выполнение функций, которые возвращают значения хранящиеся на сервере, не блокируется пока не будет получен непосредственный ответ сервера или не возникнет ошибка. Избежать этого можно, используя Xflush. Каждая команда (командная процедура) включает в себя выполнение Xflush, который посылает все запросы X серверу. Если не вызывать Xflush, то возможно непоследовательное выполнение действий, что приведёт к ошибке.

*Особенности реализации на Windows.*При использовании Windows OS эмуляция воздействий с клавиатуры и мыши осуществляется посредством функций Win32 API (Application Program Interface), которые предоставляют доступ к низкоуровневым сервисам компьютерной операционной системы.

Воздействия с клавиатуры и мыши (добавленные в Tcl команды): Передвижение курсора в заданную позицию (xw_movePointer), нажатие и удержание кнопки мыши (xw_buttonPress), отпускание кнопки мыши (xw_buttonRelease), клик кнопкой мыши (xw_buttonClick), нажатие клавиши на клавиатуре (xw_pressKey), отпускание клавиши на клавиатуре (xw_releaseKey). Для реализации этих команд использована функция Win32 API SendInput, которая последовательно посылает события в виде структур типа INPUT в поток ввода клавиатуры или мыши.

Вспомогательные пользовательские команды: Получение координат окна приложения (xw_getCoordinates), установка и снятие базовых координат (xw_setBaseCoordinates/xw_unsetBaseCoordinates), задержка прохождения теста на заданное время в миллисекундах (xw_Delay), поиск окна с заданным заголовком (xw_findWindow), перемещение окна приложения на передний план (xw_raiseWindow), активизация приложения (xw_setFocus), передвижение окна в заданные координаты (xw_moveWindow). Перечисленные команды реализованы с помощью функций работы с окнами на Win32 API Microsoft Platform Software Development Kit.

Работа с изображениями: Снятие изображения окна приложения (xw_takeSnapshot). Изображение срезается с экрана и сохраняется в BMP формате в текущей системной палитре. Использованный формат BMP является аппаратно независимым (DIB – device independent bitmap), что позволяет использовать сохраненные ранее изображения на другом компьютере, обеспечив только совместимость пользовательских настроек таких, как разрешение экрана и глубина цвета. Формат также имеет развитую программную поддержку, таким образом, упрощая разработку библиотеки команд обработки изображений. Минусом формата является несжатость хранимых на жестком диске данных. Команда формирования маски для изображения (xw_mask). Маска используется для того, чтобы исключить из последующей проверки заданные места изображения.

Сравнение изображений производится с учётом масок задающих включаемые для сравнения области и исключаемые из сравнения(xw_compareSnapshot). На вход команды подаются два BMP файла, которые загружаются в память в виде объектов типа BITMAP. Сравнение осуществляется попиксельно.

Разработка команд работы с изображениями произведена на основе функций работы с bitmap в Win32 API.