Использование автоматического тестирования пользовательского интерфейса для систем автоматизированного проектирования

Бирюков Е.Д., Копылов М.С., ИПМ им. М.В. Келдыша РАН
birukov@gin.keldysh.ru, kopylov@gin.keldysh.ru

Аннотация

Данная работа описывает различные методы автоматического тестирования графического интерфейса пользователя в программном обеспечении, в частности применительно к системам автоматизированного проектирования и дополнений (plug-ins) к ним.

1 Введение

Существуют различные виды тестирования программного обеспечения. Тестированию необходимо подвергать наличие требуемой функциональности; быстродействие имеющихся алгоритмов; их корректную работу при различных исходных данных; а также пользовательский интерфейс и его корректную реакцию на различные действия пользователя, в том числе ошибочные.

В последнее время широко распространяются системы автоматического тестирования программного обеспечения. Такие системы позволяют существенно увеличить объем тестирования при одновременном снижении усилий. Особенно эффективно использование автоматического тестирования при большом количестве однотипных тестов, например, при тестировании работы одного алгоритма с разными наборами исходных данных, или тестировании поведения нескольких однотипных диалоговых окон [Huang, Carter, 2003].

2 Необходимость применения отдельных тестов для пользовательского интерфейса

Значительную часть тестов программного обеспечения, в том числе систем автоматизированного проектирования, можно провести вообще без использования графического интерфейса пользователя. В большинстве систем автоматизированного проектирования существует возможность записи и последующего выполнения сценариев на базе того или иного языка (или специального технологического стандарта, обеспечивающего поддержку нескольких языков для написания сценариев – например, стандарта COM в системе CATIA [CATIA user manual, 2014]). Такие сценарии производят операции над объектами сцены, загруженной в систему автоматизированного проектирования, в том числе проводят рендеринг, расчеты различных физических процессов и другие вычисления, которые являются основной задачей соответствующих САПР. Результатом работы подобных операций являются растровые изображения, таблицы и другие наборы данных, которые можно сравнивать с аналогичными данными, полученными из предыдущих версий, и на основе таких сравнений давать однозначный ответ на вопрос о пригодности к работе тестируемой программы. Такой метод, называемый регрессионным тестированием, зачастую является практически полностью достаточным для тестирования систем автоматизированного проектирования и дополнений к ним, особенно тех, которые моделируют отдельные узкоспециализированные физические процессы [Волобой, Денисов, Барладян, 2014].

Отдельное тестирование пользовательского интерфейса необходимо, в первую очередь, в том случае, когда низкоуровневый набор объектов и функций (SDK) для пользовательского интерфейса пишется с нуля, либо сам интерфейс имеет сложную разветвленную структуру с большим количеством диалоговых окон, открываемых друг из друга. Однако, даже для такого программного обеспечения, которое имеет относительно простой пользовательский интерфейс, и этот интерфейс построен на базе готовых SDK, все равно могут потребоваться отдельные тесты пользовательского интерфейса. В программном обеспечении, производящем сложные математические вычисления, симулирующие различные физические процессы (к такому ПО относятся системы автоматизированного проектирования и дополнения к ним) все вы-

числительные процедуры можно разделить на две категории. К первой категории относятся расчеты, затрагивающие всю загруженную сцену или достаточно большую ее часть. Такие расчеты можно выполнить в консольном режиме с заранее заданными параметрами без использования графического интерфейса пользователя. Ко второй категории относятся расчеты, затрагивающие один или несколько отдельных объектов в сцене и предназначенные для отображения какой-либо дополнительной информации об этом объекте с помощью графического интерфейса пользователя. В их число входит, например, расчет максимального рассеяния света для заданного материала, который отображается в диалоговом окне параметров данного материала. Единственным способом инициировать подобные вычисления для проверки их правильности является открытие соответствующих элементов графического интерфейса пользователя с последующим заданием параметров.

3 Стандартные методы тестирования графического интерфейса пользователя

Все методы тестирования программного обеспечения можно разделить на две группы: метод "черного ящика", когда внутренние алгоритмы работы системы неизвестны тестировщику (или автору автоматического теста), и метод "белого ящика", когда такие алгоритмы известны, и их особенности учитываются при создании автоматического теста [Tretmans, Belinfante, 1999]. Тесты пользовательского интерфейса, как правило, относятся к первой группе, так как для пользователя, который работает с интерфейсом, обычно важен только конечный результат работы программы при различных данных, которые он вводит с помощью такого интерфейса.

Системы автоматических тестов пользовательского интерфейса могут предусматривать ручное написание сценариев тестирования на некотором встроенном языке, автоматическое создание таких сценариев в процессе выполнения самим пользователем необходимых действий с пользовательским интерфейсом, а также оба способа создания сценариев тестирования.

Для автоматического тестирования пользовательского интерфейса обычно используется система идентификаторов, которые присваиваются каждому элементу управления в интерфейсе, и команд, которые система автоматического тестирования посылает объектам с данными идентификаторами, эмулируя действия пользователя. В операционных системах семейства Windows используются дескрипторы окон (при этом под окнами понимаются любые элементы управления), и сообщения, которые можно послать окнам с соответствующими дескрипторами с помощью функции SendMessage [Memon, Banerjee, 2003]. Если каждый элемент управления однозначно реагирует на действия мыши и клавиатуры, при этом достаточно лишь чтобы указатель мыши находился над любой точкой элемента управления, то такая система автоматического тестирования обеспечивает полную эмуляцию действий пользователя. Однако отличительной особенностью систем автоматизированного проектирования при тестировании интерфейса является необходимость выделять с помощью мыши не только стандартные оконные элементы управления, но также различные объекты, отображенные в основном окне просмотра. С точки зрения операционной системы окно отображения сцены является одним элементом управления, и объекты, нарисованные в этом окне, не имеют никаких внешних идентификаторов, по которым их можно найти внешней программой. В большинстве систем автоматизированного проектирования существует то или иное отображение иерархической структуры объектов в сцене в виде дерева, обычно отображаемое в отдельном окне. Зачастую такое окно является стандартным элементом управления (например, рассматривается как TreeViewControl в операционной системе Windows), что позволяет выделять объекты в сцене с помощью такого дерева. Однако система автоматизированного проектирования CATIA отличается от других подобных систем тем, что дерево объектов сцены отображается в том же окне, что и сама сцена. Стандартное окно системы CATIA с отображенным в нем деревом сцены показано на рис. 1.

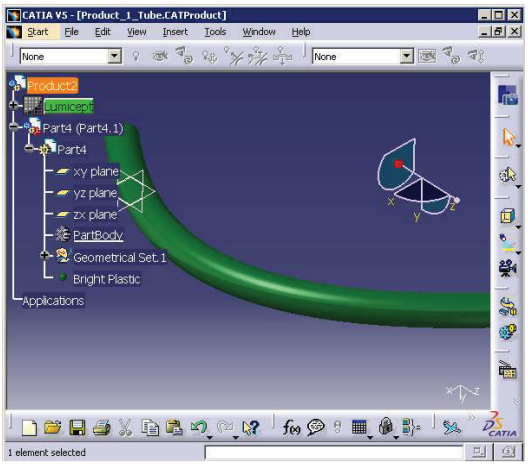

Рис. 1. Окно системы CATIA с загруженной сценой и отображенным деревом иерархии сцены

Ни само дерево, ни его отдельные элементы не имеют каких-либо внешних идентификаторов, позволяющих активировать их извне. Поэтому в любом случае необходимы некоторые дополнительные действия для выделения объектов в сцене. Кроме того, при создании дополнений к существующим системам автоматизированного проектирования иногда неизвестны идентификаторы элементов интерфейса, если внутренний исходный код их SDK является закрытым. Автоматизация тестирования пользовательского интерфейса в этом случае может быть произведена тремя способами, которые будут подробно описаны в следующем разделе.

4 Методы тестирования пользовательского интерфейса в условиях закрытого исходного кода и сложных элементов интерфейса

Первый способ выделения мышью определенной точки на элементе интерфейса предусматривает жесткое прописывание в сценарии теста конкретных координат этой точки. Основным недостатком данного метода является то, что экранные координаты, в которых размещается некоторый объект, могут меняться в зависимости от многих параметров: значений масштаба и прокрутки сцены в самой программе, а также текущего масштаба, в котором отображаются в данный момент все элементы управления на рабочем столе [Kepple, 1994]. Чтобы убедиться в том, что координаты объекта правильные с учетом текущих параметров отображения, можно перед проведением теста сбрасывать все параметры отображения в программе. Обычно такие настройки задаются через конфигурационный файл (или несколько файлов) Если автоматические тесты требуются для тестирования дополнений к существующей системе автоматизированного проектирования, то такие файлы могут иметь неизвестный формат, или даже находиться в неизвестном месте. Во втором случае сбросить параметры вообще не представляется возможным, в первом случае для этого приходится удалять весь конфигурационный файл (или несколько файлов) полностью. При этом сбрасываются не только требуемые настройки отображения, но и другие настройки, что может помешать тестированию в некоторых случаях. Что же касается настроек разрешения экрана и масштаба элементов управления операционной системы, они также хранятся либо в конфигурационных файлах, либо в специальном системном реестре. Все эти настройки также необходимо сбрасывать перед началом тестирования.

Все вышеуказанные действия существенно затрудняют любую другую работу с тестируемой программой, кроме собственно проведения автоматического тестирования пользовательского интерфейса, да и вообще могут затруднить любую другую работу на данном компьютере. Поэтому для проведения автоматического тестирования в указанном варианте рекомендуется использовать отдельное рабочее место (компьютер, дополнительная операционная система или виртуальная машина с дополнительной операционной системой) только для проведения тестирования.

Второй способ заключается в использовании систем распознавания изображений. На экране необходимо найти изображение, совпадающее с заданным образцом, определить его координаты, и эмулировать нажатие кнопки мыши в точке с этими координатами. Теоретически данный способ позволяет обеспечить независимость от разрешения экрана, масштаба элементов управления и положения окна на рабочем столе, однако при этом возникает большое количество новых проблем. Во-первых, системы распознавания изображений в настоящее время все еще не могут гарантировать стопроцентную точность срабатывания, особенно, если масштаб эталонного изображения значительно отличается от масштаба реального. Во-вторых, в окне отображения сцены может находиться несколько объектов, похожих на заданный эталон. Чтобы выделить из них конкретный объект, требуемый в данном случае, необходимо задать в

сценарии системы автоматического тестирования дополнительные параметры - подчас весьма сложные для формального кодирования. Таким образом, само написание автоматического теста заметно усложняется. Наконец, нужный объект может вообще быть не виден при начальном положении камеры в сцене (и даже соответствующий ему элемент в иерархическом дереве сцены) может быть изначально не виден - чтобы его отобразить, необходимо прокрутить сцену в сторону или дерево вниз с помощью полосы прокрутки на некоторое заранее неизвестное значение и/или развернуть один или несколько элементов в дереве, являющихся родительскими для данного. Формальное кодирование подобных действий (тривиальных для квалифицированного человека-тестировщика) еще более затруднительно, и вероятность ошибки становится слишком большой.

Третий способ, выбранный авторами в качестве основного, представляет собой комбинацию использования сценариев, встроенных в тестируемую программу, и внешней системы автоматических тестов пользовательского интерфейса.

Как правило, значительную часть действий, выполняемых проверяемой программой при тестировании пользовательского интерфейса, составляют операции над внутренними объектами, которые можно выполнить при помощи встроенной системы сценариев, без активации элементов самого пользовательского интерфейса. А действия, выполняемые только путем активации некоторых элементов пользовательского интерфейса, обычно связаны с дополнительными диалоговыми окнами, внутри которых уже корректно работают традиционные системы автоматического тестирования интерфейса. Таким образом, становится возможным сделать автоматический тест, основу которого составляет сценарий, выполняющий некоторые действия над внутренними объектами тестируемой программы, вызывающий в процессе этого открытие требуемых элементов пользовательского интерфейса (в основном - диалоговых окон), а после открытия таких элементов интерфейса - передающий управление внешней системе автоматического тестирования, которая уже будет производить действия в открытом диалоговом окне. После завершения работы внешней системы автоматического тестирования элементов пользовательского интерфейса управление снова может передаться внутреннему сценарию, который продолжит работу до следующего открытия нового элемента пользовательского интерфейса, тестирование которого опять будет выполнять внешняя программа.

4.1 Пример автоматического теста для диалогового окна параметров материала в системе CATIA

Рассмотрим для примера тестирование диалогового окна редактирования параметров материала в системе CATIA. Данное диалоговое окно является частью самой системы CATIA, однако к нему были добавлены дополнительные параметры.

Тестирование состоит из следующих этапов:

1. Запуск системы CATIA в стандартном режиме с графической оболочкой, но с автоматическим запуском сценария:

```
"C:\Program Files\Dassault Systemes\B25\win_b64\code\bin\
CNExt.exe" -env "My_CATIA_Application" -direnv
"C:\Users\User\AppData\Roaming\
DassaultSystemes\CATEnv" -macro "Macro1.catvbs"
```

Здесь параметр `-env "My_CATIA_Application"` означает название рабочего окружения системы CATIA, которое используется при тестировании (в данном случае - название нашего дополнения к системе CATIA); параметр `-direnv "C:\Users\User\AppData\Roaming\DassaultSystemes\CATEnv"` - путь к системной папке, где лежат списки всех существующих рабочих окружений для CATIA; `-macro "Macro1.catvbs"` - путь на сценарий, который будет запущен сразу после запуска CATIA.

2. Сценарий на языке CATVBs (разновидность языка Basic), который выполняет действия над внутренними объектами:

```
Sub CATMain()
Set material1 = MyWorkbench.GetMaterialByName ("Material 1")
CATIA.ActiveDocument.Selection.Add(light1)
CATIA.StartCommand "Properties"
CATIA.SystemService.ExecuteProcessus "cmd.exe /c 'test.au3'"
```

```
  If material1.MyParam = 1 Then
  CATIA.SystemService.Print
"Test passed"
  Else
  CATIA.SystemService.Print
"Test failed"
  End Sub
```

Данный сценарий производит поиск материала с заданным именем, добавляет его в список выделенных объектов, вызывает стандартную команду Properties, которая открывает диалог параметров объекта, после чего вызывает внешний инструмент для автоматического тестирования диалога параметров. Команда `SystemService.ExecuteProcessus` ждет завершения работы внешнего инструмента, на протяжении этого времени диалог параметров остается открытым. Его закрытие произойдет уже далее, с помощью внешнего инструмента. После закрытия диалога параметров управление снова переходит к данному сценарию. Появляется возможность прочитать требуемые параметры из объекта материала и убедиться, что они были правильно заданы с помощью диалога параметров.

3. Наконец, необходимо написать сценарий для работы внешнего инструмента автоматического тестирования интерфейса. В настоящей работе используется инструмент AutoIt, в котором сценарий тестирования пишется вручную. Образец написанного сценария представлен ниже:

```
  Local    $main_win    =    Win-
Wait("[REGEXPTITLE:(?i)CATIA   V5
*]", "", $Timeout)
  If $main_win = 0 Then
     Exit(1)
  EndIf
  WinActivate($main_win)
  Local    $prop_dlg    =    Win-
Wait("Properties")
  Local  $catia_pid  =  WinGetPro-
cess($main_win)
  if  WinGetProcess($prop_dlg)  <>
$catia_pid Then
     Exit(1)
  EndIf
  WinActivate($prop_dlg)
  if  ControlFocus($find_obj_dlg,
"", "SpnDifReflPos") = 0 Then
     Exit(1)
  EndIf
  Send ("0.5")
  ControlClick("[ACTIVE]",    "",
"OK")
```

Этот сценарий находит диалоговое окно с заголовком "Properties", проверяет, что это окно принадлежит системе CATIA, активирует его, затем передает фокус ввода в элемент управления (поле ввода) с именем "SpnDifReflPos" в этом диалоговом окне, записывает в него текст "0.5" и нажимает кнопку OK.

5 Выводы

Описанная схема позволяет обеспечить автоматическое тестирование большинства операций в системах автоматизированного проектирования и дополнениях к ним. Тесты пользовательского интерфейса, написанные по такой схеме, эффективно дополняют созданный ранее набор регрессионных тестов, предназначенный для проверки правильности работы основных вычислительных алгоритмов систем автоматизированного проектирования. При этом вышеописанные тесты включают в себя, в том числе, сценарии, написанные на встроенном в систему CATIA языке CATVBs, и выполняющие операции над внутренними объектами в данной САПР. Такие же сценарии используются и в обычных регрессионных тестах, которые применяются для тестирования дополнений к этой САПР. Это позволяет повторно использовать часть написанных ранее сценариев и тем самым уменьшить объем работы при создании автоматических тестов.

Система автоматических тестов пользовательского интерфейса, описанная в данной статье, в настоящее время начинает использоваться при разработке системы оптического моделирования и реалистичной компьютерной графики, которая ведется в ИПМ им. М.В. Келдыша РАН [Барладян, Дерябин, Шапиро, 2013], в особенности, той ее части, которая представляет собой дополнение к САПР CATIA.

Список литературы

Барладян Б.Х., Дерябин Н.Б., Шапиро Л.З., 2013. Интеграция модуля компьютерной графики в систему CATIA Новые информационные технологии в автоматизированных системах: материалы шестнадцатого научно-практического семинара - Моск. Ин-т электроники и математики национального исследовательского университета «Высшая школа экономики».

Волобой А.Г.,. Денисов Е.Ю, Барладян Б.Х., 2014. Тестирование систем моделирования освещенности и синтеза реалистичных изображений Программирование, № 4, 2014, с.13-22.

Dassault Systemes, Inc. CATIA Vertsion 5-6 R2015 Documentation, 2014

Zhenyu Huang, Lisa Carter, 2003. Automated solutions: Improving the efficiency of software testing. Issues in Information Systems Journal

L.R. Kepple, 1994. The black art of GUI testing. Dr. Dobb's Journal of Software Tools, 19(2):4

Atif Memon Adithya Nagarajan, 2003. GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing. In Proceedings of the 10th Working Conference on Reverse Engineering (WCRE '03)

Jan Tretmans, Axel Belinfante, 1999. Automatic Testing with Formal Methods. In EuroSTAR'99: 7th European Int. Conference on Software Testing, Analysis and Review, Barcelona, Spain