УДК 004.582

ПРЕДСТАВЛЕНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ ДЛЯ ТЕСТИРОВАНИЯ ПРОГРАММ

А.С. Пригожев

Одесский национальный политехнический университет,
65044, Одесса, проспект Шевченко, 1,
тел. 8 (048) 779-75-66, 8 (048) 779-71-06
prigozhev@matrix.odessa.ua

В работе представлено описание графического интерфейса пользователя в виде графа. Показано, что при дополнении информации о структуре интерфейса в виде графа информацией о вероятности выбора пользователем того или иного компонента управления возможно построить наиболее вероятный маршрут исполнения программы. Предложенная структура может быть использована при построении функциональных тестов программных систем.

In the article a description of the graphical user interface in the graph form is presented. Shown that the addition of information about the structure of the interface in the form of a graph of information about the probability the user selects a particular component of management is possible to construct the most likely route of program execution. The proposed structure can be used in constructing the functional tests for software systems.

Введение

Высокая сложность современных программных систем делают достаточно трудоемким процесс тестирования программного обеспечния. Поэтому сейчас актуальной является задача построения систем автоматизации тестирования программного обеспечения.

При выполнении тестирования программного обеспечения основной проблемой является полнота выполненных тестов, а также критерий окончания тестирования. Наиболее распространенным критерием сейчас является исчерпание времени, отводимого на этот процесс [1]. Однако при использовании такого критерия, очевидно, существует риск обнаружения невыявленных ошибок в процессе эксплуатации. В случае выявления подобного рода ошибок, а также для их предупреждения, программа как правило направляется на регрессионное или повторное тестирование. При данном виде тестирования программа проходит не только стандартный набор тестов, но и составленный с помощью анализа отзывов пользователей. Однако проведение данного анализа часто вызывает определенные сложности, поскольку зачастую отзывы пользователей носят неформализованный характер и описывают действия пользователя лишь приближенно.

Поэтому важными являются задачи разработки формализованного представления действий пользователя при работе с интерфейсом, особенно графическим, а также представление ответов программы на указанные действия пользователя. Такая формализация позволила бы в значительной степени облегчить анализ работы пользователя, сформировать наиболее вероятные маршруты исполнения программы в процессе работы пользователя, а также автоматизировать процессы повторного и регресионного тестирования.

Существующие средства тестирования пользовательских интерфейсов

Среди существующих на сегодняшний день средств тестирования пользовательского интерфейса можно выделить WinRunner, QA Run, Silk Test, VisualTest и Robot [2, 3]. Рассмотрим наиболее интересные системы.

Visual Test – комплексная среда автоматизации тестирования от компании Rational, позволяющая тестировать самый широкий спектр приложений. Это достаточно развитая система автоматизированного тестирования для приложений операционных систем Microsoft Windows 95, Windows NT, а так же приложений, предназначенных для работы в WWW. Visual Test позволяет моделировать всевозможные ситуации и подвергать информационные системы практически любого масштаба критическим испытаниям. Сценарии тестирования могут задаваться в интерактивном режиме и с помощью специального рекордера записываться на диск, а затем использоваться многократно для повторного тестирования. Visual Test независим от средств разработки и может применяться для тестирования систем, созданных с использованием любых языков программирования. С помощью Visual Test можно тестировать не только готовые системы, но и отдельные программные компоненты. Недостатком данной системы является отсутствие возможности тестирования программ, ориентированных на другие операционные системы.

Borland SilkTest – инструмент автоматизации процесса функционального тестирования корпоративных прикладных программ через графический интерфейс пользователя (GUI). Программа представляет собой решение для регрессионного, кросс-платформенного и локализационного тестирования в широком диапазоне технологий разработки приложений, в том числе Web, Java, .NET и клиент-серверных технологий в рамках современных коротких циклов тестирования. Инструмент SilkTest предназначен для реализации преимуществ автоматизации даже при использовании сложных контрольных примеров и предлагает набор средств для повышения производительности, которые позволяют без лишних сложностей работать с изменениями в тестируемых приложениях. Кроме того, мощная инфраструктура тестирования обуславливает высокую воспроизводимость тестовых сценариев в тестируемых проектах.

SilkTest обращается к тестируемому приложению точно так же, как это делает реальный пользователь – через графический интерфейс пользователя, благодаря чему достигается тестирование методов работы конечного пользователя в полном объеме. Чтобы сделать возможным выполнение тестов даже на нескольких ком-пьютерах (в распределенной среде), SilkTest моделирует действия пользователей через отдельный агентский компонент. Поскольку этот агент занимает лишь небольшое пространство в памяти, он легко размещается на удаленных клиентах. Недостатком решения является слабое развитие средств моделирования действий пользователей на основе протокола работы.

Rational Robot является решением для автоматизированного тестирования, которое позволяет, единожды написав тест, вторично использовать его на любых платформах. Технология Object Testing, используемая в Rational Robot, позволяет всесторонне тестировать приложения, что особенно важно в современных средах разработки, где производительность разработки ПО с использованием встроенных компонентов резко возросла. Эти различные библиотеки классов Java, средств управления ActiveX Control, OLE Control (OCX), Visual Basic Control (VBX), объекты Visual Basic, Win32 Control и т. д.

Rational Robot генерирует сценарии функциональных тестов на языке SQABasic, который синтаксически подобен обычному Visual Basic. С помощью SQABasic можно просматривать и редактировать сценарии тестов прямо во время записи. Rational Robot включает в себя встроенный редактор и отладчик с режимом анимационного воспроизведения и онлайновой проверкой синтаксиса скрипта. В любое время можно дополнить сценарии тестов какими-либо процедурами и логическими условиями. При этом доступен вызов любой функции из DLL или из API-интерфейса Windows.

Постановка задачи исследования

На основе проведенного анализа можно сделать вывод о необходимости дальнейшего развития средств автоматизированного тестирования пользовательского интерфейса в направлении независимости от языка программирования и использования статистики деятельности пользователей в программной системе. Для решения указанной задачи представляется целесообразным использование иерархических сценариев [4]. Данная структура данных нуждается в определенной модификации, для представления в ее терминах структуры пользовательского интерфейса и моделирования деятельности пользователей в интерфейсе. Цель настоящего исследования – разработка унифицированной модели для автоматизированного языконезависимого тестирования програмных систем с графическим интерфейсом.

Представление графического интерфейса в виде сценариев

Основной структурной единицей графического интерфейса является окно. Существует несколько основных типов окон графического интерфейса:

- окна ввода данных;
- информационные окна с множественным выбором;
- окна сообщений об ошибках.

Пример окна ввода данных показан на рис. 1.

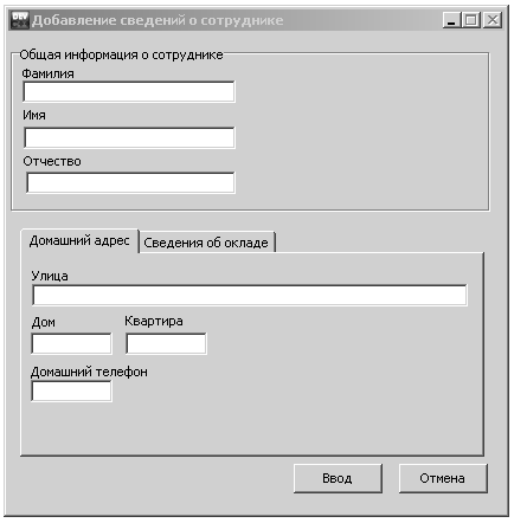

Рис. 1. Окно ввода данных

Информационные окна ввода данных, как правило, имеют иерархическую структуру и состоят из двух основных типов компонентов: компоненты ввода данных, информационные компоненты, компоненты-контейнеры. К компонентам ввода данных относятся разнообразные поля ввода, списки, выпадающие списки и кнопки. К числу элементов-контейнеров относятся элементы типа группа и набор закладок, которые позволяют логически сгруппировать элементы управления в окне. Компонент типа группа отличается от компоненты панель закладок тем, что в данном компоненте, как правило, отсутствуют обработчики событий, связанных с вводом данных. Напротив, эта группа событий может присутствовать в панели закладок и определяться для каждой закладки в отдельности. В итоге группа информационных компонентов также статична, и обычно включает в себя разнообразную текстовую информацию.

Структуру окна ввода данных, показанного на рис. 1, можно представить в виде дерева, фрагмент которого для приведнного фрагмента интерфейса показан на рис. 2.

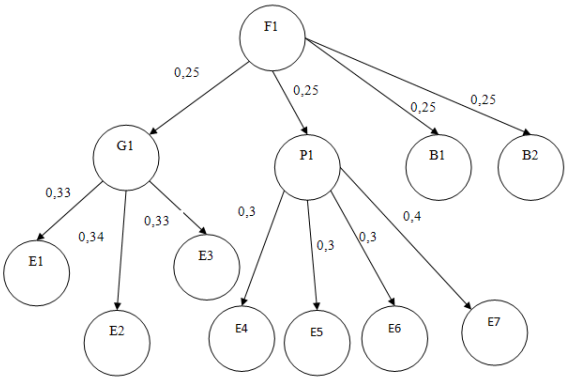

Рис. 2. Дерево структуры интерфейса

Вершинами дерева являются компоненты управления, находящейся на форме. Корневой вершиной является сама форма, вершины второго уровня и последующих уровней (кроме листовых вершин) соответствуют элементам контейнерам, а листовые вершины – непосредственно информационными элементами. На графе, изображенном на рис. 2 вершины E1-E7 соответствуют полям редактирования окна, показанного на рис. 1, вершина B1 – кнопке «Ввод», вершина B2 – кнопке «Отмена», вершина G1 обозначает групповой элемент с заголовком «Общая информация о сотруднике», вершина P1 – первой странице многостраничного контейнера «Домашний адрес». На рис. 2 не показана вершина графа P2 соответствующая закладке «Информация об окладе».

Поскольку любая программа с графическим интерфейсом функционирует на основе событий, сопоставленных компонентам управления, то для того, чтобы выяснить какой маршрут исполнения программы является наиболее вероятным, необходимо знать вероятности использования того или иного компонента в рамках пользовательского интерфейса, поскольку зная эту информацию, легко определить какой из обработчиков событий наиболее часто выполняется. На графе (см. рис. 2) вероятности использования того или иного компонента графического интерфейса показаны на ребре, конечной вершиной которого является вершина, соответствующая данному компоненту.

Код обработчиков событий в интерфейсе представляется в виде графа [4]. Вероятность выполнения данного обработчика вычисляется как произведение вероятности выбора компонента на вероятность появления этого события в компоненте. При этом полную группу событий образуют события связанные с выбором любого компонента управления на форме, а также все события с выбранным компонентом, для которых определены обработчики указанных событий.

Такой подход позволяет автоматизировать поиск наиболее вероятного маршрута исполнения программы. Начальной точкой такого маршрута может быть выбран, например, компонент ввода данных, имеющий наименьшие координаты в окне. Окончанием маршрута может являться компонент, выполнение одного из событий которого приводит к закрытию окна.

Аналогичными деревьями могут быть представлены и другие две группы окон, с тем отличием, что соответствующее дерево структуры интерфейса будет иметь одну корневую и несколько листовых вершин.

Представленный на рис. 2 граф используется в качестве унифицированной модели пользовательского интерфейса для языконезависимой среды тестирования программ [5].

Использование такого подхода, наряду с ранее рассмотренными алгоритмами анализа окна позволяют выявить наиболее часто генерируемые пользователем последовательности событий, и проанализировать их код без привязки к конкретному языку программирования. Существующие средства мониторинга деятельности пользователя, описанные, например в [6], позволяют в значительной степени автоматизировать указанный процесс.

Выводы

При проведении исследования проанализированы существующие системы автоматизации тестирования пользовательских интерфейсов. В результате анализа установлено, что существующие средства недостаточно универсальны, в частности не обеспечивают поддержку тестирования программ для различных операционных систем, а также не позволяют определить наиболее вероятный маршрут исполнения программы.

Для дальнейшего развития указанных средств предложено использовать граф структуры интерфейса в качестве основы построения среды автоматизации регрессионного и повторного тестирования. Для формализации поведения пользователя в данной системе предложено использовать вероятность выбора того или иного компонента пользователем. Показано, что использование такой модели позволяет выявить наиболее вероятные пути выполнения программы.

Для дальнейших исследований в данной области предполагается разработать формальные методы анализа построенной модели, с целью получения наборов значений для тестирования программы.

1. *Андон Ф.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю.* Основы инженерии качества программных систем. – 2-е изд., перераб и доп. – К.: Академпериодика, 2007. – 672 с.
2. *URL*: http://www.interface.ru
3. *Автоматическая* генерация тестов для графического пользовательского интерфейса по UML диаграммам действий [Електронний ресурс] / Калинов А.Я., Косачев А.С., Посыпкин М.А., Соколов А.А. // Режим доступу: http://www.citforum.ru/SE/testing/generation_uml/
4. *Пригожев А.С.* Автоматизированный перевод исходных текстов программ // Електромашинобудування та електрообладнання. – 2009. – № 72. – С. 222–225.
5. *Пригожев А.С.* Языконезависимая среда разработчика для тестирования программного обеспечения // Радіоелектронні та комп'ютерні системи. – 2009. – № 7. – С. 225–230.
6. *Пригожев А.С.* Реализация решателя и базы знаний экспертной системы для планирования действий пользователя при разработке программ // Радиоэлектроника и информатика. – 2005. – № 4. – С. 110–116.