СОЗДАНИЕ ОБЪЕКТНОЙ МОДЕЛИ ДЛЯ ТЕСТИРОВАНИЯ ИНТЕРНЕТ ПРИЛОЖЕНИЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Мычко А.И.

Одинец Д. Н. – кандидат. техн. наук, доцент

Автоматизация тестирования в области web-интерфейса пользователя предполагает создание программы для моделирования действий пользователя. В работе рассмотрено создание объектной модели на основании которой будут создаваться автоматизированные тесты.

Автоматизация тестирования – та часть приложения, которая отвечает за стабильность его разработки. Качество приложения зависит от количества автоматизированных тестов. Чем их больше, тем сложнее их поддерживать: исправлять ошибки, делать изменения.

Существуют следующие автоматизированные тесты:

1. Автосгенерированные - могут быть созданы с помощью различных программ.
2. Написанные с помощью специальных программ - нужно писать вручную на специфическом языке специальной программы.
3. Написанные с помощью языка программирования. Принцип их работы основан на моделировании действий пользователя с помощью специальных библиотек (например Selenium). Но они требуют написание автотестов с нуля. Этот вид тестов является самым интересным, но при этом этот способ самый тяжелый.

Такие тесты требуют применения дополнительных паттернов для более эффективного их использования. Одним из самых эффективных паттернов является Page object (создание тестов с разделенной логикой, когда логика теста отделена от примитивных действий над тестируемым приложением). Паттерн очень эффективен, но создание таких тестов создает наложение большого количества дополнительных затрат: перед началом написания таких тестов необходимо написать модель тестируемой системы, которая будет совершать нужные действия над приложением.

Создание такой модели должно в себя включать несколько этапов. Программа, автоматизирующая создание тестов, должна уметь находить нужные для автоматизации страницы, уметь распознавать на этих страницах все доступные элементы, переходить на другие страницы, путем воздействия на элементы интерфейса и все это представлять в объектной модели выбранного языка программирования. Для перехода по страницам и нахождения на них элементов интерфейса пользователя, а также для манипуляций с этими элементами можно задействовать проект Selenium. Во время перехода весь путь обхода доступных страниц будет преобразовываться в модель конечного автомата, где каждая новая страница будет представлена в виде состояния автомата, а элементы интерфейса будут переходами, которые позволяют попасть на другую страницу (в другое состояние). Каждый переход должен быть задействован и установлено состояние, к которому он приведет. В итоге получается граф переходов конечного автомата, по которому можно построить модель классов на выбранном языке программирования. При построении автомата может быть так, что переход вызовет состояние, которое не будет нужным (например, страница другого сайта). Для отбрасывания таких состояний алгоритм фильтровать все сайты по нужному доменному имени. Также при переходе нужно помечать, что переход уже проработан и еще раз нет необходимости по нему переходить. Но такой подход не даст выявить всех возможных состояний, при всех различных комбинациях условий для переходов, но дает выигрыш во времени создания тестируемой модели. Для нахождения остальных состояний в программе должны быть специальные настройки, какие позволяют переключиться в режим построения полного автомата с выявлением всевозможных переходов. По полученному графу переходов строится модель для тестирования приложения: строятся классы для тестируемой системы - имена всех локаторов выносятся в текстовый файл, а все похожие страницы выделяются в суперклассы. Каждый класс должен содержать в виде атрибутов все доступные ему переходы. Эти атрибуты должны быть доступны пользователю с помощью свойств класса. Кроме этих атрибутов в классе должны содержаться ссылки на другие классы (состояния), попасть в которые мы может с помощью одного из переходов данного класса. При воздействии на атрибут, ссылающийся на другой класс, активным должен становиться тот класс, на который мы сослались. Таким образом классы связываются между собой в целостную модель, предоставляющую возможность управлять тестируемой системой. При нахождении одинаковой сигнатуры классов, но при различных значениях внутри сигнатуры должен быть задействован полиморфизм. Также использование полученного графа переходов позволит моделировать тесты, планируя их таким образом, чтобы пройти все переходы.

Благодаря приведенной методике можно создать программу, позволяющую автоматизировать процесс создания объектной модели тестируемой веб-системы. Основываясь на результат работы программы можно создавать непосредственно автоматизированные тесты.

Список использованных источников:

1. Gerard Meszaros, xUnit Test Patterns: Refactoring Test / Gerard Meszaros, Pearson Education, Boston, 2007. - 883 p.
2. F.Cohen, Java Testing and Design: From Unit Testing to Automated Web Tests / F. Cohen, Prentice Hall PTR, 2004. – 544 c.
3. Марченков С. С., Конечные автоматы / Марченков С. С., ФИЗМАТЛИТ Москва, 2008. - 56 с.