Journal of Siberian Federal University. Engineering & Technologies 2022 15(2): 274–282
~ ~ ~

DOI: 10.17516/1999-494X-0390

УДК 519.6

Architecture of Intelligent Testing System

Vladimir A. Boyko*[a], Alexander I. Legalov[b] and Sergey V. Zykov[b]
[a]Siberian Federal University
Krasnoyarsk, Russian Federation
[b]Higher school of economics
Moscow, Russian Federation

Received 06.07.2021, received in revised form 18.11.2021, accepted 15.03.2022

Abstract. One of the approaches to automating graphical user interface testing is software systems that reproduce the scenario of user interaction with the software under test. The process of playing such a scenario is based on the ability to simulate user actions. The principles underlying the implementation of such a testing system can have a significant impact both on the reliability of the reproduction of test scenarios and on the degree of integration of the system into the software under test or the environment that runs it. The overwhelming majority of tools are not able to provide high reliability and portability of test scripts. At the same time, the problem of ensuring the quality of work and testing various user interfaces, regardless of the platform that launches it, screen size, and input method, is extremely complex, multifaceted, and still not effectively solved. The modular architecture of an intelligent graphical user interface testing system is considered. To achieve greater autonomy of the intelligent system, various implementation options have been proposed. A brief description of intellectualization based on machine learning methods is given. The elements of the novelty of the presented architectural solution are modular structure, cyclic execution of test script commands, and the presence of an intelligent module capable of recognizing elements of the graphical user interface. On the basis of the concept of intellectualization of the graphical user interface testing process, the modular cyclical architecture of an intelligent testing system has been created and the process and features of the interaction of intelligent system modules are described. The concept of the test script command execution cycle is proposed and approaches to ensuring the autonomy of an intelligent system are considered. The proposed system can be used to solve problems of integration testing, using intellectualization based on machine learning methods, an intelligent system with a modular architecture allows you to achieve a high level of testing reliability.

Keywords: testing, intelligent systems, machine learning, modular structure, graphical user interface.

Citation: Boyko, V.A., Legalov, A.I., Zykov, S. V. Architecture of Intelligent Testing System, J. Sib. Fed. Univ. Eng. & Technol., 2022, 15(2), 274–282. DOI: 10.17516/1999-494X-0390

© Siberian Federal University. All rights reserved
This work is licensed under a Creative Commons Attribution-Non Commercial 4.0 International License (CC BY-NC 4.0).

* Corresponding author E-mail address: ryt161@mail.ru

Архитектура интеллектуальной системы тестирования

В. А. Бойко[а], А. И. Легалов[б], С. В. Зыков[б]
[а]Сибирский федеральный университет
Российская Федерация, Красноярск
[б]Высшая школа экономики
Российская Федерация, Москва

Аннотация. Одним из подходов к автоматизации тестирования графического интерфейса пользователя являются программные комплексы, воспроизводящие сценарий взаимодействия пользователя с тестируемым программным обеспечением. Процесс воспроизведения такого сценария основывается на возможности имитации действий пользователя. Принципы, заложенные в реализацию такой системы тестирования, могут оказывать значительное влияние как на достоверность воспроизведения сценариев тестирования, так и на меру интеграции системы в тестируемое программное обеспечение или запускающую его среду. Подавляющее большинство инструментов не способно обеспечить высокой достоверности и переносимости тестовых сценариев. Вместе с тем, проблема обеспечения качества работы и тестирования различных интерфейсов пользователя, вне зависимости от запускающей его платформы, размеров экрана и способа ввода, является чрезвычайно сложной, многогранной и до сих пор эффективно не решенной. Рассматривается модульная архитектура интеллектуальной системы тестирования графического интерфейса пользователя. Для достижения большей автономности интеллектуальной системы предложены различные варианты реализации. Дано краткое описание интеллектуализации, основанной на методах машинного обучения. Элементами новизны представленного архитектурного решения служат модульная структура, циклическое выполнение команд тестового сценария и наличие интеллектуального модуля, способного распознавать элементы графического интерфейса пользователя. На основе концепции интеллектуализации процесса тестирования графического интерфейса пользователя создана модульная циклическая архитектура интеллектуальной системы тестирования и описан процесс и особенности взаимодействия модулей интеллектуальной системы. Предложена концепция цикла выполнения команды тестового сценария и рассмотрены подходы к обеспечению автономности работы интеллектуальной системы. Предложенная система может применяться для решения задач интеграционного тестирования с использованием интеллектуализации, основанной на методах машинного обучения; интеллектуальная система с модульной архитектурой позволяет достичь высокого уровня достоверности тестирования.

Ключевые слова: тестирование, интеллектуальные системы, машинное обучение, модульная структура, графический интерфейс пользователя.

Цитирование: Бойко, В. А. Архитектура интеллектуальной системы тестирования / В. А. Бойко, А. И. Легалов, С. В. Зыков // Журн. Сиб. федер. ун-та. Техника и технологии, 2022, 15(2). С. 274–282. DOI: 10.17516/1999-494X-0390

Введение

На сегодняшний день тестирование является одним из ключевых элементов разработки программного обеспечения (ПО), гарантирующим доступность различных частей приложения, функциональность и стабильность. Тестирование графического интерфейса пользователя (ГИП) – одна из задач интеграционного тестирования. Оно позволяет проверить как работоспособность продукта, так и количество действий или, другими словами, число шагов пользователя для достижения результата. Существуют инструментальные средства, направленные

на автоматизацию данного процесса. Очевидна целесообразность их использования в процессе разработки любого программного обеспечения, имеющего ГИП.

Специфика работы данных инструментов основана на имитации действий пользователя программного обеспечения. Эти действия можно описать с применением сценариев, определяющих последовательность перемещений курсора мыши, нажатий на ее клавиши, ввода данных с клавиатуры, а также функций, определяющих взаимодействие с компьютером посредством других внешних устройств. Инструмент для тестирования ГИП, в свою очередь, реализует описанные взаимодействия, используя встроенные модули, имитирующие работу внешних устройств в соответствии с заданным сценарием.

Запуская автоматические тесты, тестировщики ПО сталкиваются с рядом проблем. Например, интерфейс пользователя может зависеть от настроек отображения, изменяемых средствами операционной системы, не связанными с разрабатываемым программным продуктом. Данная проблема возникает при тестировании ПО с использованием различных оконных менеджеров и разрешений экрана [1]. Проблемы с тестированием ГИП могут возникать и при отсутствии средств, обеспечивающих кроссплатформенное тестирование. Это приводит к значительному увеличению затрат на адаптацию имеющихся тестов под разные платформы [2, 3]. Следствием указанных ограничений является ситуация, при которой многие решения в процессе тестирования предлагается возлагать на тестировщика.

В связи с отсутствием аппарата для межплатформенного ввода и обработки информации актуальна задача разработки инструмента тестирования ГИП, использующего интеллектуализацию, основанную на методах машинного обучения [11, 12] для тестирования ГИП в независимой от платформы вычислительной среде и проведения тестирования без прямой связи с проверяемым приложением.

В работе предлагается новый подход к тестированию ГИП, основанный на комбинации интеллектуального распознавания элементов ГИП и последующей имитации действий пользователя путем воздействия на них через доступные устройства ввода, и реализация программного решения на основе данного подхода.

Цель работы – создание модульной интеллектуальной системы, позволяющей работать с различными операционными системами, размерами экранов и стилей ГИП. Интеллектуальная система с такой модульной архитектурой способна обеспечить высокий уровень достоверности тестирования.

1. Концепция интеллектуальной системы тестирования

Интеллектуальная система тестирования – это программно-аппаратный комплекс, позволяющий выполнять сценарии взаимодействия с ПО, на основе использования интеллектуальных подходов.

Проведенный анализ [4] показывает, что в основе работы ИСТ должны лежать следующие принципы:

1) независимость от внутреннего устройства тестируемого ПО;

2) переносимость сценариев между реализациями, исполняемыми в разных системах, размерах экранов, разрешений и т. д.;

3) адаптивность для использования с различными типами шрифтов, цветами, отступами и другими изменчивыми свойствами ГИП.

Процесс тестирования с использованием ИСП представлен use case диаграммой (рис. 1).

Система позволяет тестировщику загрузить сценарий тестирования, на основе которого система запускает цикл тестирования. Тестирование производится через постоянное сканирование экрана тестируемого ПО, распознавание на нем элементов ГИП и взаимодействие с этими элементами через симуляцию нажатий реального пользователя. По завершении сценария система предоставляет тестировщику отчет о проведенном тестировании и завершает работу.

Сценарий тестирования для ИСТ представляет собой набор команд, описывающих взаимодействие пользователя и тестируемого ПО. В таком сценарии опускаются любые параметры, характеризующие конкретную платформу, внутреннее устройство тестируемого ПО или параметры отображения ГИП. Данный подход позволяет сценарию оставаться кроссплатформенным, независимым и переносимым [4].

Написание тестовых сценариев для ИСТ позволяет достичь принципа – один тест для всех версий тестирующего ПО, который отвечает концепции кроссплатформенного тестирования. Однажды написанный сценарий не требует дальнейшей корректировки для различных вариаций шрифтов, визуальных эффектов и прочего, задачи по работе с этими параметрами отныне ложатся на интеллектуальные алгоритмы распознавания, обеспечивающие отсутствие зависимости от настроек отображения.

Как видно из описания системы, тестировщик при работе с ней освобождается от значительного количества задач, связанных с определением побочных параметров тестируемого

Рис. 1. Use case диаграмма работы ИСТ

Fig. 1. Use case diagram of ITS

объекта. Сценарий, создаваемый для ИСТ, можно выполнить, задав лишь тип объекта и его название. Такой подход не требует от тестировщика знаний о внутреннем устройстве тестируемого ПО, это означает, что ИСТ независима от исходного кода программы.

Малый объем входных данных и отсутствие возможности взаимодействия с исходным кодом тестируемой системы означают необходимость решить следующие задачи:

– задача поиска элементов ГИП – ИСТ необходимо различать элементы ГИП, такие как кнопки, и уметь определять их местоположение для взаимодействия;

– задача взаимодействия с элементами ГИП – взаимодействие с тестируемым ПО производится исключительно через устройства ввода, например клавиатура и мышь, и устройства вывода, например экран монитора, отображающий визуальное представление тестируемого ПО или ГИП;

– задача управления процессом взаимодействия – для управления ИСТ необходим сценарий, позволяющий описывать взаимодействия с тестируемым ПО без учета особенностей отдельных платформ.

Такая система позволяет полностью смоделировать работу обычного пользователя ПО без доступа к исходному коду тестируемого ПО.

2. Архитектура интеллектуальной системы тестирования

Архитектура ИСТ состоит из взаимодействующих между собой систем и модулей, способных включать в себя несколько систем. Части системы разделены следующим образом:

– модуль чтения ГИП;

– модуль сценариев тестирования;

– модуль обработки инструкций;

– система управления элементами ГИП.

Взаимодействие модулей ИСТ между собой происходит при выполнении команд тестового сценария. Текстовый сценарий состоит из набора последовательных действий, выполнение таких действий будет производиться в цикле. Тестирование с использованием ИСТ является циклом последовательного получения команд, содержащихся в сценарии, выполнения их и анализа полученного результата.

Перед запуском цикла выполнения сценария необходимо получить актуальное изображение тестируемого ПО от модуля чтения ГИП. После получения такого изображения модуль сценариев тестирования может переходить к выполнению команды тестового сценария.

Команда передается из модуля управления сценарием в модуль обработки инструкций, кроме того, в него передается и актуальное состояние тестируемого ПО в виде изображения.

Модуль обработки инструкций, используя алгоритмы компьютерного зрения, находит координаты элемента ГИП с указанным в команде типом и названием.

Получив координаты элемента ГИП, ИСТ готова к взаимодействию с тестируемым ПО. Координаты отправляются в систему управления элементами ГИП. Основываясь на типе взаимодействия, указанного в команде, ИСТ имитирует действия пользователя.

В данной работе предлагается схема работы ИСТ (рис. 2), опирающаяся на использование автономной системы распознавания состояния тестируемого ПО непосредственно с экрана мо-

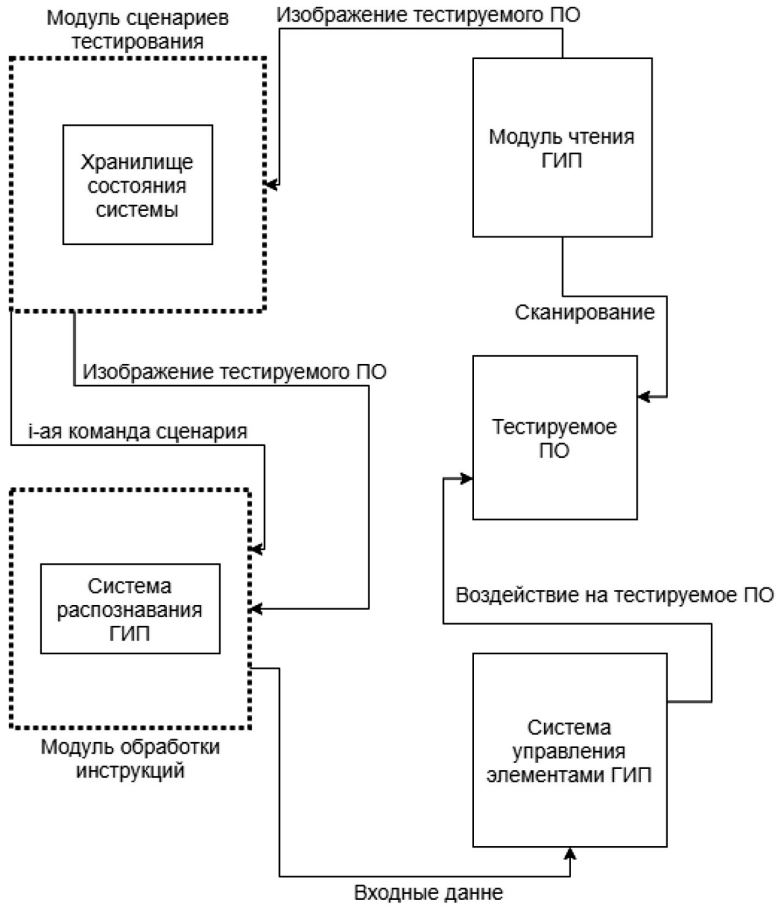

Рис. 2. Схема взаимодействия модулей ИСТ

Fig. 2. Scheme of interaction of ITS modules

нитора, которая через средства эмуляции клавиатуры и мыши обеспечивает управление тестируемым ПО.

Написание тестовых сценариев для ИСТ позволяет достичь принципа – один тест для всех версий тестирующего ПО, который отвечает концепции кроссплатформенного тестирования. Однажды написанный сценарий не требует дальнейшей корректировки для различных вариаций шрифтов, визуальных эффектов и прочего, задачи по работе с этими параметрами отныне ложатся на интеллектуальные алгоритмы распознавания, обеспечивающие отсутствие зависимости от настроек отображения.

Тестировщик при работе с системой освобождается от значительного количества задач, связанных с определением побочных параметров тестируемого объекта.

3. Модуль чтения графического интерфейса пользователя

Модуль чтения ГИП – это модуль, сканирующий актуальное состояние окна тестируемого программного обеспечения, на котором отображаются элементы тестируемого ГИП. Актуальное состояние окна тестируемого ПО представляется в виде изображения.

Модуль чтения ГИП сканирует экран и сверяет полученное изображение с текущим состоянием, в случае нахождения различий новое состояние экрана отправляется в модуль сценариев тестирования.

Получение визуальных данных, представляющих актуальное состояние тестируемого ПО, может быть произведено:

1) как захват экрана – при таком подходе ИСТ должна быть запущена на одной платформе с тестируемым ПО, что не позволяет в полной мере имитировать работу пользователя;

2) как съемка экрана платформы, на которой запущено тестируемое ПО, – такой подход имитирует работу пользователя, который видит перед собой экран тестируемого ПО.

4. Модуль сценариев тестирования

Этот модуль запускает чтение нового теста и накапливает тесты в хранилище. Модуль отвечает за запуск и выполнение команд, описанных в сценарии тестирования, заключение вывода о соответствии актуального состояния тестируемого ПО ожидаемому состоянию и хранение общего состояния ИСТ. Получая актуальное состояние экрана тестируемого ПО и информацию о наличии ошибок, модуль, основываясь на текущем сценарии, может продолжить выполнение сценария, сделать вывод по сценарию или запустить следующую команду сценария.

5. Модуль обработки инструкций

Модуль обработки инструкций – это модуль ИСТ, выполняющий поступающие команды сценария и опирающийся в своей работе на методы машинного обучения [11, 12], используемые для поиска и детектирования элементов ГИП. Получая экран тестируемого, система поиска элементов ГИП, используя изображение экрана, вычисляет координаты данного элемента при помощи нейронной сети. В результате модуль формирует данные, необходимые для взаимодействия с целевым элементом ГИП, и отправляет их далее по циклу.

6. Система управления графическими элементами пользователя

Система управления элементами ГИП – это система, производящая манипуляции с платформой, запускающей тестируемое ПО. Такие манипуляции могут производиться через использование устройств ввода, таких как мышь и клавиатура.

Используя систему управления элементами ГИП, ИСТ может управлять тестируемым ПО через беспроводные (Bluetooth) или проводные (USB) каналы.

Правильное управление элементами ГИП достигается за счет знания их координат и типов, так, например, нажатие кнопки и выбор элемента выпадающего списка требуют совершенно разного набора действий.

Воздействие на тестируемое ПО может быть произведено одним из следующих способов:

1. Программно, через интерфейсы доступа – ИСТ запускается на одной платформе с тестируемым ПО, в результате чего получает возможность воздействовать на устройства ввода через программные интерфейсы доступа.

2. Подключение к физическим интерфейсам доступа – ИСТ запускается на отдельном компьютере, который соединяется с компьютером, на котором запущено тестируемое ПО по-

средством подключения к физическим интерфейсам и отправки через них команд, имитирующих команды, посылаемые клавиатурой и мышью.

3. Роботизированные руки – использование роботизированной техники позволит ИСТ не взаимодействовать напрямую с компьютером, на котором запущено тестируемое ПО, вместо этого ИСТ будет подавать команды роботу, который, в свою очередь, используя клавиатуру и мышь, будет взаимодействовать с тестируемым ПО.

Заключение

В статье предложена архитектура интеллектуальной системы тестирования, процесс интеллектуализации в которой основан на использовании нейронной сети. Реализация подобной архитектуры позволяет обеспечить независимость от настроек отображения тестируемого ПО, кроссплатформенность, независимость от исходного кода тестируемого ПО и возможность обратной связи в процессе выполнения тестового сценария. Также стоит отметить, что представленный подход может быть использован не только в процессе тестирования ПО, но и в дистанционном образовании [9, 10] как система автоматизации выполнения рутинных или повторяющихся задач.

Список литературы / References

[1] Соловьева А. А. Сравнение программного обеспечения для разработки пользовательских интерфейсов и их прототипирования, *Наука без границ*, 2020, 4 (44), 55–60. [Soloveva A. A. Comparison of software for developing and prototyping user interfaces, *Science without borders 2020*, 4 (44), 55–60 (in Russian)]

[2] Deshpande A.P., Mahender C. N. Summarization of graph using question answer approach, *Advances in Intelligent Systems and Computing 2020*, 933, 205–216.

[3] Jander M. A multiple-choice protocol tester, *Data Communications*, 1994, 23(7), 41–42.

[4] Бойко В. А. Анализ особенностей процесса тестирования графического интерфейса пользователя, *Приборы и системы. Управление, контроль, диагностика*, 2021, 2. [Boyko V. A. Analysis of the features of the graphical user interface testing process, *Devices and systems. Management, control, diagnostics*, 2021, 2 (in Russian)]

[5] Рассел С., Норвиг П. *Искусственный интеллект: современный подход*, 2-е изд.: Пер. с англ. М.: Издательский дом «Вильямс», 2007. 1408 с. [Russell S., Norvig P. *Artificial Intelligence: A Modern Approach*, 2nd ed. M.: Publishing house «Williams», 2007. 1408 p. (in Russian)]

[6] Саттон Р.С., Барто Э. Г. *Обучение с подкреплением*: Пер. с англ. М.: БИНОМ. Лаборатория знаний, 2011. 399 с. [Sutton R. S. Barto E. G. *Reinforcement learning*. M.: BINOM. Knowledge Laboratory, 2011. 399 p. (in Russian)]

[7] Шайтура С. В. *Интеллектуальные системы и технологии*. Бургас, 2016. [Shaytura S. V. *Intelligent systems and technologies*, Burgas, 2016 (in Russian)]

[8] Минитаева А.М., Пешков Д. В. Нейронная сеть для определения принадлежности к заданному языку, *Славянский форум*, 2019, 1 (23), 174–181. [Minitaeva A. M., Peshkov D. V. Neural network for determining belonging to a given language, *Slavic Forum*, 2019, 1 (23), 174–181 (in Russian)]

[9] Shaitura S.V., Ordov K. V., Pigoreva O. V., Kosterina I. V., Zyukin D. A., Gerasimova V. G. Problems of distance education, *Revista Inclusiones*, 2020, 7(S4–1), 24–38.

[10] Shaytura S.V., Ordov K. V., Minitaeva A. M. Digital learning methods for the digital economy, *Proceedings of the International Scientific and Practical Conference on Digital Economy (ISCDE2019)*, Yekaterinburg, Russia, pp. 606–611 doi:10.2991/iscde-19.2019.117.

[11] Chen, Jieshan & Xie, Mulong & Xing, Zhenchang & Chen, Chunyang & Xu, Xiwei & Zhu, Liming. Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination? *Proceedings of the Sixth International Conference on Green and Human Information Technology* 2020, 86–90.

[12] Rahmadi, Agyl & Sudaryanto, Aris. Visual Recognition Of Graphical User Interface Components Using Deep Learning Technique. *Jurnal Ilmu Komputer dan Informasi*, 2020, 13(1), 35.