DOI 10.53364/24138614_2022_25_2_101
УДК 378.1

[1]**Буравов А.А.,** [2]**Дузбаев Н. Т.**
[1,2]Международный университет информационных технологий, РК, Алматы

[1]E-mail: 24793@iitu.edu.kz
[2]E-mail: n.duzbayev@iitu.edu.kz

СРАВНЕНИЕ ПОДХОДОВ К АВТОМАТИЗИРОВАННОМУ ТЕСТИРОВАНИЮ ПРОГРАММНОГО КОДА В ОНЛАЙН-ОБРАЗОВАНИИ

АВТОМАТТАНДЫРЫЛҒАН ТЕСТІЛЕУ ТӘСІЛДЕРІН САЛЫСТЫРУ ОНЛАЙН БІЛІМ БЕРУДЕГІ БАҒДАРЛАМАЛЫҚ КОД

COMPARISON OF APPROACHES TO AUTOMATED TESTING PROGRAM CODE IN ONLINE EDUCATION

Аннотация. При создании образовательного контента в сфере изучения языков программирования и информационных технологий качественная автоматизация процесса проверки практических заданий студентов позволяет существенно улучшить качество обучения. Существует множество подходов к автоматизированному тестированию и проверке кода студентов, каждый из которых обладает своими особенностями, достоинствами и недостатками. В данной статье проведены сравнительный анализ и классификация существующих подходов к автоматической проверке практических заданий в образовательных курсах по информационным технологиям.
Ключевые слова: МООК, онлайн-курсы, тестирование, онлайн-обучение, автоматическое тестирование.

Abstract. High-quality automation of the process of checking students' practical tasks can significantly improve the quality of education in the field of studying programming languages and information technologies. There are many approaches to automated testing and verification of student code, each of which has its own characteristics, advantages and disadvantages. This article provides a comparative analysis and classification of existing approaches to automatic verification of practical tasks in educational courses on information technologies.
Keywords: MOOC, online courses, testing, online education, autograding.

Аңдатпа. Бағдарламалау тілдерін және ақпараттық технологияларды оқыту саласында білім беру мазмұнын құру кезінде студенттердің практикалық тапсырмаларын тексеру процесін жоғары сапалы автоматтандыру білім сапасын айтарлықтай жақсартуға мүмкіндік береді. Студенттік кодты автоматтандырылған тестілеу мен тексерудің көптеген тәсілдері бар, олардың әрқайсысының өзіндік сипаттамалары, артықшылықтары мен кемшіліктері бар. Бұл жұмыста ақпараттық технологиялар бойынша білім беру курстарындағы практикалық тапсырмаларды автоматты түрде тексерудің қолданыстағы тәсілдерін салыстырмалы талдау және жіктеу қарастырылған.
Түйін сөздер: ЖАОК, онлайн курстар, тестілеу, онлайн оқыту, Автоматты тестілеу.

Введение. В настоящее время онлайн-обучение становится все более важным аспектом образования в сфере информационных технологий (ИТ), так как растут потребности общества в разработчиках, тестировщиках и других специальностях, а классический формат

обучения часто не позволяет быстро адаптироваться к появляющимся новым специализациям и технологиям [1]. Особенно это стало очевидно с началом пандемии COVID-19 [2]. При этом невозможно разработать качественный курс по информационным технологиям без подробного практического компонента. Одним из самых долгих процессов, узких горлышек процесса дистанционного ИТ-образования является этап тестирования и проверки практических заданий студентов [3]. Существуют различные способы автоматизировать данный этап, начиная от применения существующих открытых решений, заканчивая разработкой собственного автогрейдера с тестами API, интерфейса и других выходных каналов коммуникации с веб-приложением [4].

Целью данной статьи является классификация, а также сравнительный анализ существующих в настоящее время подходов к автоматизации тестирования приложений, разработанных студентами в рамках образовательных решений по информационным технологиям.

Обзор литературы. В данном разделе представлен краткий обзор существующих исследований и научных работ по данной проблеме.

Штаубиц и др. [5] описывают некоторые типовые сценарии взаимодействия пользователей с онлайн-курсами. Авторы показывают базовые низкоуровневые интеграции и некоторые части архитектуры решений с функционалом проверки кода, однако не приводят подробных практических примеров.

Глассман и др. [6] рассматривают различные подходы к просмотру кода и построению логического дерева тестов. В данной работе показаны примеры решений по подробному статистическому анализу проверки студенческого кода. Можно отметить, что авторы затрагивают некоторые аспекты автоматического тестирования приложений с графическим интерфейсом, однако без описания технической архитектуры.

Киралья и др. [7] описывают разработку платформы для онлайн-курсов с функционалом автоматической проверки кода на Java с использованием тестовой среды. Также в данной работе рассмотрены некоторые методы построения системы оценки на основе тестовых библиотек и фреймворков.

Дерваль и др. [8] описали архитектуру и принцип открытой платформы автоматизированной проверки заданий под названием INGInious. Данная платформа работает с контейнерами Docker, обеспечивая независимость логики от языка программирования, на котором написан код студента. Она также предлагает дополнительные функции такие как редактирование кода в браузере, подключение к курсам edX в качестве внешнего автогрейдера. Однако в качестве недостатка описанной реализации можно отметить отсутствие функционала проверки заданий на разработку графического интерфейса (фронтенда).

Hundt, Schlarb и Schmidt [9] представляют веб-приложение для автоматизированной оценки заданий по параллельному программированию. Достаточно подробно описан аспект создания системы с распределенными экземплярами для эмуляции реальной сети для последующего тестирования решений студентов.

Робинсон и Кэрролл [10] описывают платформу онлайн-обучения с открытым исходным кодом, которая помимо обратной связи от автогрейдера позволят выводить студенту также индивидуальные контекстные рекомендации преподавателя по заданию.

Канас и др. [11]описывают открытую платформу RoboticsAcademy. Данная платформа предназначена для дистанционного обучения студентов навыкам в области робототехники. Образовательный контент, разработанный авторами, включает в себя специальный скриптовый язык работы с робототехникой, (ROS) задания на взаимодействие с 3D-симулятором движения робота и задания на работу с языком программирования Python. Можно отметить описание интеграции платформы с потенциальными реальными объектами и компонент системы, отвечающий за автоматическую оценку загруженного студентом скрипта управления дроном.

Манзур и др. [12] описывают применение открытой платформы Web-CAT для автоматической оценки студенческих решений, написанных в Jupyter Notebooks. Разработанное авторами расширение Jupyter Notebook позволяет учащимся напрямую загружать исходные файлы с кодом для последующей проверки в Web-CAT. Авторы достаточно сжато описали техническую часть и особенности интеграции Web-CAT, Canvas и расширений Jupyter Notebook.

Сияфудин и др. [13] описывают разработанную платформу APLAS для обучения программированию для Android. Платформа включает в себя валидатор (автогрейдер), веб-интерфейс и базу данных, а также несколько учебных курсов. Данная работа примечательна своей специализированной направленностью на автоматизированное тестирование приложений на Android и имеет уникальную техническую специфику.

Барлоу и др. [14] представляют обзор наиболее популярных решений по автоматизированной проверке кода, доступных в настоящее время, а также описывают разработанное ими веб-приложение MOCSIDE, являющееся масштабируемой онлайн-средой разработки с открытым исходным кодом и функционалом автоматической проверки заданий студентов по информатики. Примечательно то, что авторы избрали подход контейнеризации с помощью Docker в качестве основного алгоритма автогрейдера. Однако, в данной статье слабо описана техническая часть, архитектура и алгоритмы представленной системы.

Классификация подходов к автоматизированной проверке

В настоящее время при автоматизации тестирования кода можно выделить следующие подходы (тестирующий модуль здесь и далее называется автогрейдером):

A. Компиляция и коммуникация со скомпилированным приложением по стандартным каналам ввода-вывода (stdin-stdout), запуск консольного приложения. Данный метод является одним из старейших и простых подходов к тестированию решений, написанных на любом языке программирования. Его основным достоинством является простота реализации. Недостатки: необходимость настраивать компиляцию и сборку для каждого языка программирования, сложность при реализации сложных сценариев и тесткейсов, необходимость учитывать множество пограничных сценариев в логике автогрейдера (например, появление ошибки или исключения в тестируемом приложении).

B. Файловый ввод-вывод. При использовании данного метода автогрейдер компилирует решение, используя файлы с определенными адресами и названиям в качестве входных данных для тестируемого приложения, аналогично считывая файлы для проверки выходных данных. Этот метод можно использовать в сочетании с предыдущим подходом (A). Применение файлового ввода-вывода позволяет тестировать работу с большими объемами данных, чем при работе через стандартные потоки ввода-вывода (A).

C. Тестирование веб-API. При таком подходе тестируемое приложение собирается и запускается как веб-сервер (бэкенд). Далее автогрейдер отправляет запросы для тестирования API по протоколам TCP, HTTP, HTTPS, WebSocket, сравнивая ответы приложения с ожидаемыми. Можно отметить такие достоинства метода, как простоту, меньшую зависимость от языка. Однако такой подход применим не во всех случаях, достаточно часто возникает необходимость тестировать также веб-интерфейс, либо другие каналы коммуникации с приложением. Кроме того, для написания приложения с web-API студент должен уже иметь навыки написания веб-сервера, а это обычно уже ближе к продвинутому уровню владения языком программирования.

D. Тестирование с помощью встроенных инструментов языка. Большинство языков программирования имеют собственные инструменты для написания внутренних тестов. Например, Jest для JavaScript, тестовые библиотеки для Golang, NUnit для .NET, и т.д. При таком подходе преподаватель должен сперва написать автоматические тесты с помощью выбранной библиотеки. Преимущества его использования — широкая поддержка функций языка, возможность быстрого написания тест-кейсов. Недостатки — необходимость заново

писать тесты для каждого языка программирования, а также требование к навыкам преподавателя, т.е. нельзя поручить написание тестовых сценариев ассистенту [15].

E. Тестирование графического интерфейса. Данный подход становится необходим, когда студенты работают над задачами с графическим интерфейсом. Это могут быть веб-страницы, настольные или мобильные приложения. В рамках этого подхода автогрейдер может проверять HTML-теги на отображаемой веб-странице, сравнивать цвета пикселей в определенных координатах, эмулировать нажатия клавиш и щелчки мыши пользователем. Преимущества - независимость от языка программирования, так как работа ведется с интерфейсом. Недостатки — сложность настройки тесткейсов, наличие множества возможностей обмануть автогрейдер, например, отобразив части интерфейса специальным образом [7].

F. Контейнеризация. При использовании этого подхода приложение упаковывается в контейнер (достаточно часто применяется Docker). Далее автогрейдер может взаимодействовать с запущенным приложением, применяя подходы A, B, C, E — взаимодействовать через консоль, работать с файлами, делать веб-запросы, проверять статические html-файлы либо графический интерфейс. Достоинствами данного метода являются отсутствие необходимости модификации среды и тестовой логики для каждого языка программирования, возможность комбинировать другие подходы. Недостатки — повышенное потребление ресурсов по сравнению с остальными средствами и возможные ошибки при остановке и удалении контейнеров [16].

Нами представляется наиболее удобным способ классификации вышеперечисленных подходов по критерию привязки/зависимости от конкретного языка программирования. Можно выделить три группы подходов:

• Полная привязка к конкретному языку программирования или фреймворку;

• Независимость от технологии на этапе запуска тесткейсов, но зависимость (разные настройки) на этапе компиляции и запуска приложения;

• Практически полная независимость от языка программирования.

Классификация по данному критерию приведена на Рис. 1.

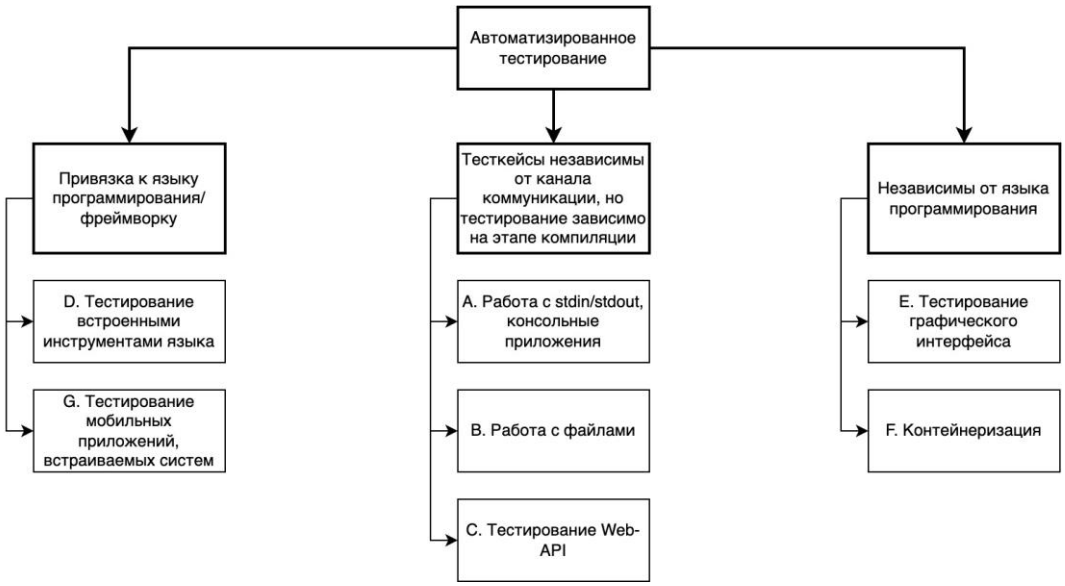

Рисунок 1. Классификация подходов к автоматизированному тестированию

Сравнение подходов к автоматизированной проверке

В рамках данной работы нами был проведен сравнительный анализ вышеперечисленных подходов к автоматизированному тестированию кода студентов. Сравнение производилось по следующим критериям:

R1 – независимость от языка программирования на этапе компиляции (сборки) тестируемого приложения. Оценивается как 1 – да и 0 – нет.

R2 – независимость от языка программирования на этапе выполнения тесткейсов. Оценивается как 1 – да и 0 – нет.

R3 – возможность сложной настройки тесткейсов, например, добавление условных конструкций, циклов, вывод индивидуальных сообщений/предупреждений для студента. Оценивается как 1 – присутствует и 0 – отсутствует.

R4 – возможность полной изоляции тестируемого приложения без необходимости запускать его на отдельном сервере либо виртуальной машине. Данный критерий может быть критичен как с точки зрения безопасности, так и с точки зрения удобства разработки и настройки автогрейдера. Так как если тестируемое приложение имеет полный доступ к серверу (хосту), то после его работы в системе могут остаться уязвимости и артефакты работы (файлы, измененные системные настройки и т.д.). Оценивается как 1 – присутствует и 0 – отсутствует.

Для каждого из описанных в предыдущем разделе подходов после простановки значений критериев производилось вычисление суммарного индекса как:

Итого = R1 + R2 + R3 + R4

Результаты произведенного сравнительного анализа приведены в Таблице 1.

Таблица 1

Сравнение существующих подходов к автоматизированной проверке кода

Подход	**R1**	**R2**	**R3**	**R4**	**Итого**
A. Работа с stdin/stdout	0	1	0	0	1
B. Работа с файлами	0	1	0	0	1
C. Тестирование API	0	1	1	0	2
D. Инструменты языка	0	0	1	0	1
E. Тестирование приложений, встраиваемых систем	0	0	1	1	2
F. Тестирование графики	0	1	0	0	1
G. Контейнеризация	1	1	0	1	3

Сравнив все подходы, можно подытожить, что по сумме всех критериев наилучшие подходы это: контейнеризация, тестирование API, тестирование мобильных приложений и встраиваемых систем. При этом нужно иметь в виду, что при выборе подхода на этапе реализации новой образовательной платформы, либо улучшения существующего, нужно принимать во внимание все особенности каждого подхода, так как каждый имеет свои достоинства и недостатки для определенных условий. Например, при выборе можно руководствоваться следующими критериями:

- Количество языков программирования, которые должна поддерживать платформа;
- Необходимость тестирования API, графического интерфейса, консольного ввод-вывода;

- Как часто нужно редактировать тесткейсы (например, для обновления при выходе новой версии языка) и достаточно ли ресурсов преподавателя для этого;
- Требования к вычислительным ресурсам и быстродействию;
- Требования к безопасности и изоляции тестируемых заданий;
- Квалификация разработчиков платформы;
- Необходимость редактирования сложной логики в тесткейсах

Вывод. В рамках данной работы нами были подробно рассмотрены и описаны существующие подходы к автоматизированному тестированию кода на платформах онлайн-образования.

1. Основываясь на существующих исследовательских работах и решениях, были выделены следующие основные подходы к тестированию: консольный ввод-вывод, файловое взаимодействие, тестирование API, инструменты языка, тестирование мобильных приложений и встраиваемых систем, проверка графического интерфейса, контейнеризация.

2. Далее выделенные подходы были разделены на три классификационные группы по принципу независимости от конкретного стека технологий (языка программирования, фреймворка, используемых библиотек).

3. Для составления рейтинга и сравнительного анализа описанные подходы сравнивались по нескольким критериям, включающим в себя привязку к технологическому стеку на разных этапах жизненного цикла тестируемого решения студента, сложность настройки тесткейсов, возможность изоляции приложения.

4. Проведенными нами анализом, классификацией и ранжированием подходов можно руководствоваться при выборе алгоритма тестирования при создании новой образовательной платформы, или доработке существующей. При этом необходимо принимать во внимание специфику окружения и требований заказчика.

Список использованной литературы

1. Borisov V. V., Y. S.P., M. T.V., O. D.S., "Software package for managing the training of IT specialists SkillsForYou», v. 22, pp. 177–185, June, 2020, doi: 10.15827/0236-235X.130.177-185.

2. Patricia Aguilera-Hermida, «College students' use and acceptance of emergency online learning due to COVID-19», International Journal of Educational Research Open, v. 1, p. 100011, 2020, doi: 10.1016/j.ijedro.2020.100011.

3. Luchaninov D. V., Bazhenov R. I., Dimitriev A. P., and Kizyanov A. O., "Using an automated programming training system for organizing independent work of students", vol. 8, ed.5, p. 11, 2020.

4. Uravov Alexey and Duzbayev Nurzhan, "Using containerization for automated testing of software in online education", Universum: technical sciences: electron. scientific. Journal, ed. 98, May 2022, [Online]. Available at: https://7universum.com/ru/tech/archive/item/13725.

5. Staubitz T., Klement H., Renz J., Teusner R., and Meinel C., «Towards practical programming exercises and automated assessment in Massive Open Online Courses», 2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), Zhuhai, China, december, 2015, pp. 23–30. doi: 10.1109/TALE.2015.7386010.

6. Glassman L., Scott J., Singh R., Guo P. J., and Miller R. C., «OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale», ACM Trans. Comput.-Hum. Interact., vol. 22, ed. 2, pp. 1–35, april, 2015, doi: 10.1145/2699751.

7. Király, Nehéz K., and. Hornyák O, «Some aspects of grading Java code submissions in MOOCs», Research in Learning Technology, vol. 25, July, 2017, doi: 10.25304/rlt., v.25, 1945.

8. Derval G., Gego A., Reinbold P., Frantzen B., and Roy P. V., «Automatic grading of programming exercises in a MOOC using the INGInious platform», p. 6, 2015.

9. Hundt C., Schlarb M. and Schmidt B., «SAUCE: A web application for interactive teaching and learning of parallel programming», Journal of Parallel and Distributed Computing, vol. 105, pp. 163–173, July, 2017, doi: 10.1016/j.jpdc.2016.12.028.

10. Robinson P. E. and Carroll J., «An Online Learning Platform for Teaching, Learning, and Assessment of Programming», p. 10, 2017.

11. Bertrand S., Marzat J., Besnerais G. L., Manzanera A., Maniu C. S., and. Makarov M, «Integrating Experimental Data Sets and Simulation Codes for Students into a MOOC on Aerial Robotics», IFAC-PapersOnLine, vol. 52, ed. 9, pp. 50–55, 2019, doi: 10.1016/j.ifacol.2019.08.123.

12. Manzoor H., Naik A., Shaffer C. A., North C., and Edwards S. H., «Auto-Grading Jupyter Notebooks», в Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Portland OR USA, february, 2020, pp. 1139–1144. doi: 10.1145/3328778.3366947.

13. Syaifudin Y. W. and others «Web application implementation of Android programming learning assistance system and its evaluations», IOP Conf. Ser.: Mater. Sci. Eng., v. 1073, ed. 1, p. 012060, february, 2021, doi: 10.1088/1757-899X/1073/1/012060.

14. Barlow M., Cazalas I., Robinson C., and Cazalas J., «MOCSIDE: an Open-source and Scalable Online IDE and Auto-Grader for Introductory Programming Courses», p. 10.

15. Akahane Y., Kitaya H., and. Inoue U, «Design and evaluation of automated scoring Java programming assignments», 2015, IEEE/ACIS 16 th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Takamatsu, June, 2015, pp. 1–6. doi: 10.1109/SNPD.2015.7176255.

16. Maicus E., Peveler M., Patterson S., and Cutler B., «Autograding Distributed Algorithms in Networked Containers», в Proceedings of the 50th ACM Technical Symposium on Computer Science Education, Minneapolis MN USA, february, 2019, pp. 133–138. doi: 10.1145/3287324.3287505.

DOI 10.53364/24138614_2022_25_2_107
УДК 004.738:378

Хажиахмет Т.Н., магистрант
Научный руководитель: **Дузбаев Н.Т.,** проректор по цифровизации и инновациям PhD, асс. профессор
Международный университет информационных технологий, Алматы, РК.

[1]E-mail: tima17.1995@gmail.com
[2]E-mail: n.duzbayev@iitu.edu.kz

РАЗРАБОТКА ETL-СИСТЕМЫ ДЛЯ ЗАГРУЗКИ ДАННЫХ В ХРАНИЛИЩЕ ДАННЫХ

ДЕРЕКТЕР ҚОЙМАСЫНА МӘЛІМЕТТЕРДІ ЖҮКТЕУ ҮШІН ETL ЖҮЙЕСІН ҚҰРУ

DEVELOPMENT OF AN ETL SYSTEM FOR UPLOADING DATA TO A DATA WAREHOUSE

Аннотация. В данной статье описана основная идея разработки ETL-системы для загрузки данных в Хранилище Данных. Представлены основные задачи разработки, а также описан процесс реализации ETL-системы.

Ключевые слова: разработка ETL-системы, Хранилище Данных, BI, СУБД, API, ODI, HTTP Basic Authentication, Target, SAP BO, DMZ.