DOI: 10.15514/ISPRAS-2025-37(3)-21

Тестирование в жизненном цикле автоматизированных систем

Б.А. Позин, ORCID: 0000-0002-0012-2230 <bpozin@ec-leasing.ru>

Институт системного программирования РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.
Научно-исследовательский университет Высшая школа экономики,
Россия, 101978, Мясницкая ул., д.20.
ЗАО ЕС-лизинг, Россия, 117405, Москва, Варшавское ш. д.125.

Аннотация. Тестирование ПО автоматизированных систем на разных стадиях их жизненного цикла отличается по целям, решаемым задачам, объектам, методам и результатам тестирования, несмотря на отличие свойств этих объектов. При этом в научно-технической литературе используются одни и те же термины для описания разнотипных объектов, несмотря на различие их свойств и методов работы с ними в процессе тестирования. Целью работы являются рассмотрение используемого в жизненном цикле ПО автоматизированных систем комплекса понятий в области тестирования и изменение смыслового содержания этих понятий в зависимости от того, какими свойствами должно обладать ПО АС на текущем этапе ЖЦ. Соответственно изменяются характеристики тестируемого ПО, объемы тестов, степень соответствия объекта тестирования и необходимых для конкретного вида тестирования ресурсов для его применения. Понимание этих отличий существенно влияет на используемые виды и методы тестирования, а также на требования к средствам автоматизации тестирования.

Ключевые слова: тестирование программного обеспечения; виды тестирования; функциональное тестирование; нагрузочное тестирование; тестирование безопасности; статический анализ; композиционный анализ; динамический анализ; фаззинг-тестирование.

Для цитирования*:* Позин Б.А. Тестирование в жизненном цикле автоматизированных систем. Труды ИСП РАН, том 37, вып. 3, 2025 г., стр. 303–310. DOI: 10.15514/ISPRAS–2025–37(3)–21.

Благодарности: Автор признателен А.К. Петренко и В.В. Кулямину за плодотворные дискуссии.

Testing in Life Cycle of Automated Systems

B.A. Pozin, ORCID: 0000-0002-0012-2230 <bpozin@ec-leasing.ru>

Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
National Research University, Higher School of Economics, 20, Myasnitskaya ulitsa, Moscow, 101978, Russia
«EC-leasing» Co., 125 Varshavskoye shosse, Moscow, 117405, Russia

Abstract. Software testing of automated systems at different stages of their life cycle (LC) differs in terms of goals, tasks, objects, methods, and test results, despite the difference in the properties of these objects. At the same time, the same terms are used in the scientific and technical literature to describe different types of objects, despite the difference in their properties and methods of working with them during testing. The purpose of the work is to consider the complex of concepts used in the software lifecycle of automated systems in the field of testing and to change the semantic content of these concepts depending on what properties the automated system software should have at the current stage of the LC. Accordingly, the characteristics of the software under test, the volume of tests, the degree of compliance of the test object and the resources necessary for a specific type of testing for its application change. Understanding these differences significantly affects the types and methods of testing used, as well as the requirements for test automation tools.

Keywords: software testing; types of testing; functional testing; load testing; security testing; static analysis; compositional analysis; dynamic analysis; fuzz testing.

For citation: Pozin B.A. Testing in the life cycle of automated systems. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 3, 2025, pp. 303-310 (in Russian). DOI: 10.15514/ISPRAS-2025-37(3)-21.

Acknowledgments: The author is grateful to A.K. Petrenko and V.V. Kulyamin for fruitful discussions.

1. Введение

Программное обеспечение (ПО) может быть как самостоятельным продуктом (операционная система, сервис), так и представлять собой составную часть системы, реализующую ее функции в интересах решения некоторой задачи автоматизации (управления бизнесом, транспортом, финансовой, коммерческой деятельностью и т.п.).

Для контроля качества человеко-машинных программно-аппаратных систем (далее – систем) одним из наиболее широко применяемых на практике является метод тестирования. Тестирование представляет собой проверку соответствия реального поведения системы требованиям к ней, выполняемую на выбранном определенным образом наборе сценариев работы этой системы.

Суть тестирования состоит в том, чтобы в результате прогона (выполнения) тестируемой программы обнаружить существующий в ней дефект (успешное тестирование), то есть несоответствие объекта тестирования требованиям к нему. Тестирование – это прежде всего обнаружение ошибки. Однако этот метод можно использовать и для контроля качества. Разделяют понятия тестирование и отладка. Отладка включает *обнаружение* дефектов (собственно тестирование); *локализацию* мест, в которых дефекты/ошибки расположены в тестируемой программе; диагностики *ошибок* (определение типов ошибок и целей их устранения); *исправление ошибок* – установление фактов, подтверждающих исправление ошибок, и того, что ПО соответствует требованиям к нему.

Следует особенно отметить, что процесс тестирования осуществляется на всех стадиях жизненного цикла системы: при разработке, эксплуатации, сопровождении и развитии ПО систем.

2. Требования к системе и ПО как эталоны для тестирования

Требования, которые предъявляются к системе в целом и к ее ПО, формулируются в документах, чаще всего в техническом задании (ТЗ) на систему и в конкретизирующих его частных технических заданиях (ЧТЗ) или спецификациях [1, 2], согласованных Заказчиком. При проектировании, разработке ПО и системы в целом эти требования обязательны к выполнению при разработке. Требования технического задания являются требованиями высокого уровня: функциональными и нефункциональными и используются при тестировании. При тестировании ПО автоматизированных систем (АС) они являются эталоном для проверки соответствия ПО и системы (объекта тестирования) предъявляемым требованиям (эталону для тестирования).

Функциональные требования представляют собой описание того, какие функции должны быть выполнены, и иногда описывают исходные данные и ожидаемые результаты выполнения системой или ПО функций. Нефункциональные требования описывают интересующие заказчика системы нефункциональные свойства, такие как производительность, надежность, безопасность и другие, зачатую с указанием уровня достижимых значений этих характеристик. Конкретизация значений нефункциональных требований и возможных свойств реализуемых функций в ходе разработки системы производится в техническом проекте или формализованных соглашениях (протоколах и т.п.) сторон, которые также можно интерпретировать как требования.

Совокупность требований можно проверять поэтапно, по мере реализации программных компонент на этапах разработки. При декомпозиции технического задания на стадии проектирования обычно выделяют функциональные подсистемы, каждая из которых конкретизирует требования к одной системной функции, специфицированной в ТЗ. Именно эти требования являются затем эталонными при тестировании этой системной функции.

Ключевыми понятиями (функционального) тестирования являются понятия тестовой ситуации и теста. Выбранный сценарий для проверки некоторой функции (функционального требования) называют тестовой ситуацией.

На каждом этапе разработки производят тестирование того ПО, которое реализует каждую из подсистем и соответственно проводят функциональное тестирование подсистемы, как объекта разработки, для которого существует набор функциональных требований как эталон для тестирования.

Тестированию может подвергаться как отдельная компонента программного обеспечения, так и несколько компонент. В последнем случае используют термин *интеграционное тестирование*, то есть проверку того, что несколько компонент функционируют согласованно, совместно реализуя некоторую системную функцию, предусмотренную ТЗ или ЧТЗ. Степень полноты проверки оценивается критерием полноты, который может устанавливаться как для программного обеспечения в целом, так и для проверок отдельных компонент. Во многих случаях используют понятие критерий покрытия, который изначально применялся только при тестировании структуры отдельных программных модулей, чтобы оценить, какая доля логики программ проверена тестовым набором (существовал даже стандарт BBC Великобритании по критериям покрытия логики программ [3]).

Под *системным тестированием* в процессе разработки понимают тестирование ПО системы в статике для проверки функционирования системы в определенных режимах функционирования, например, подготовка к выполнению запланированного процесса обработки данных и т.п.

При тестировании ПО на стадии разработки, начиная с интеграционного тестирования, объем тестов становится довольно большим, трудоемкость их хранения, внесения изменений и сопровождения становится значительной.

3. Виды тестирования

Существует довольно много видов тестирования, то есть подходов к построению тестовых наборов разного назначения – с учетом разных типов объектов тестирования, информации для выбора тестовых ситуаций и возможности подачи тестов на вход объекта тестирования.

По возможности использования структуры ПО в критериях покрытия различают виды тестирования, основанные на методах:

- черного ящика, когда разработчик тестов практически не имеет сведений о структуре тестируемого ПО или его компонент;
- белого ящика, когда разработчик знает внутреннюю структуру объекта тестирования и может ее использовать при разработке теста и оценке степени покрытия;
- серого ящика (смешанное), когда внутренняя структура объекта тестирования частично может быть использована.

Эффективность применения разных видов тестирования зависит от стадии жизненного цикла и используемого метода функционального тестирования. По видам тестирования на стадии ввода в действие [5, 6] различают (по отечественной практике) три вида *приемочного тестирования*:

- *предварительные испытания* для *верификации* соответствия ПО функциональным требованиям;
- *опытная эксплуатация (валидация),* целью которой является оценка соответствия системы /ПО потребностям заказчика на период проверки;
- *приемочные испытания (acceptance testing)*.

3.1 Регрессионное тестирование

После внесения изменений необходимо убедиться в том, что изменения совместимы с ранее проверенным ПО, то есть после их внесения в этом ПО не проявляются новые ошибки – ошибки регрессии. Для целей такой проверки после внесения изменений осуществляют проверку ПО на тех же тестах (тестовых наборах), на которых ПО уже проверялось ранее и подтвердило работоспособность. Такой вид тестирования называется регрессионным тестированием и имеет целью обнаружение ошибок регрессии. Регрессионное тестирование в технологически зрелых организациях проводится систематически, повторно при изменении ранее отработанного ПО – с какой бы целью эти изменения не производились. Регрессионное тестирование применяется на стадии разработки при системном тестировании для контроля совместимости обновлений с уже проверенной частью ПО и, как правило, проводится систематически (например, после каждой пересборки релиза ПО), на стадии сопровождения при проверке корректности вносимых изменений в действующую программную систему и т. п.

3.2 Методы тестирования нефункциональных требований

Нефункциональные требования предъявляются к общим свойствам системы, то есть к тому, какой должна быть система в целом: производительность, надежность, доступность, мобильность, масштабируемость и др. Объектом тестирования является система в целом. В силу этого все методы тестирования нефункциональных требований оказываются весьма дорогими при реализации, требуют тщательного планирования. Тестирование качества реализации этих требований осуществляется разными методами. Цели тестирования нефункциональных требований обычно устанавливаются для каждого эксперимента отдельно с учетом того, какие свойства системы подвергаются тестированию и как предполагается использовать результаты. По результатам тестирования нефункциональных требований оцениваются эксплуатационные характеристики системы (на момент тестирования).

Предметами тестирования являются конкретные свойства системы: при тестировании переносимости – переносимость между какими аппаратными платформами проверяется и оценивается; возможность переноса программного обеспечения из одной операционной среды в другую (конкретные среды) и т. п. При тестировании совместимости – совместимость в части конкретных протоколов и интерфейсов системы со смежными системами. При тестировании надежности – аспекты надежности, которые требуется оценить, например, тестирование времени восстановления работоспособности системы после отказа.

3.3 Нагрузочное тестирование системы

Наиболее подробно исследованным является, пожалуй, нагрузочное тестирование (performance testing), поскольку оно отвечает на важнейший вопрос: какова производительность системы в терминах количества работ (которые она автоматизирует) в единицу времени. Например, количество банковских транзакций в течение операционного дня или количество проданных билетов в час и т.п.

При нагрузочном тестировании исследуются различные свойства производительности: собственно производительность системы при плановой нагрузке; время отклика системы на запрос (среднее или по видам запросов); нагрузочная способность при резких изменениях нагрузки (soak testing); способность системы долго выдерживать повышенную нагрузку (spike testing) и др. Обычно рассматривают три вида характеристик производительности: реактивность, продуктивность и использование.

Нагрузочное тестирование – это всегда проведение нагрузочного эксперимента, тщательное планирование которого позволяет получить корректные результаты и ограничить затраты на его проведение. Для планирования нагрузочного эксперимента необходимо совместно с владельцем системы сформулировать цель проведения нагрузочного тестирования и сценарий нагрузочного эксперимента, исходя из потребностей владельца по оценке или прогнозированию эксплуатационных характеристик тестируемой системы и соответствующих нефункциональных требований, а также исходя из потребностей периодического контроля деградации эксплуатационных характеристик тестируемой системы. Все характеристики системы оцениваются методами теории вероятностей и математической статистики по результатам эксперимента, обеспечивающего по возможности достаточное количество измерений для достоверной оценки этих характеристик. В зависимости от целей нагрузочного тестирования можно выделить следующие его виды и назначение [6]:

- *Оценочное* – оценка характеристик производительности в однократном нагрузочном эксперименте;
- *Аналитическое* – выявление зависимостей (например, производительности от вычислительных ресурсов) в серии нагрузочных экспериментов;
- *Настроечное* – настройка и оптимизация нагрузочных характеристик системы или ее составной части;
- *Регрессионное* – многократное периодическое нагрузочное тестирование при неизменных условиях для выявления признаков деградации тестируемой системы.

4. Тестирование с целью поиска уязвимостей в программах

В последние годы в связи с активным использованием в разрабатываемом ПО компонент из Open Source библиотек и соответствующем возрастании риска наличия уязвимостей в ПО АС, в промышленности применяются новые методы поиска уязвимостей. Это далеко не всегда методы, похожие на классическое тестирование. В настоящее время активно используются методы статического анализа программ [8, 9], методы композиционного анализа, методы динамического анализа [10, 13].

При использовании метода *статического анализа* [5] осуществляется сопоставление конструкций анализируемой (тестируемой) программы с эталоном, каковым являются компоненты классификаций уязвимостей: OWASP, CWE, CVE, БДУ ФСТЭК. Механизм этих методов довольно понятный: проводится анализ исходных текстов ПО АС на используемом языке программирования, в процессе которого выделяется конструкция с признаками наличия уязвимости, которая сопоставляется с элементами вышеприведенных классификаций. В случае совпадения такая конструкция оценивается как уязвимость, в противном случае – как правильная конструкция[11]. Отчеты о результатах статического анализа предоставляются разработчикам ПО для доработки/внесения изменений в ПО, а также уполномоченным сотрудникам службы информационной безопасности для оценки качества разработанного ПО с позиций безопасности, а также для поддержки принятия решения о возможности промышленного использования ПО, имеющего в своем составе уязвимости того уровня критичности, который установлен по результатам статического анализа. Инструментальные средства статического анализа записывают в файл SARIF сведения о найденных уязвимостях, их уровне серьезности и рекомендации по исправлению. Подробнее о SARIF см. ниже в этом разделе.

При использовании методов композиционного анализа объектом тестирования является состав ПО АС, которое планируется поставлять в составе АС. Объект тестирования состоит из множества компонент, часть из которых написана разработчиками ПО АС для реализации функциональных задач АС, а другая часть (по имеющимся данным до 70-90%) заимствуется из библиотек Open Source. В результате возникает коллизия, состоящая в том, что авторы ПО зачастую точно не знают, какие компоненты входят в состав разработанного ими ПО, а также не до конца понимают, каков уровень безопасности заимствованных компонент и ПО в целом, то есть какие уязвимости присутствуют в составе ПО, а также в процессе разработки и отладки ПО АС- какие уязвимости исправлены, а какие нет. В том числе авторы заимствованных компонент могут и сами исправлять уязвимости, о чем в библиотеках хранится информация. Количество компонент при этом может достигать несколько сотен и более. Средства автоматизации композиционного анализа проводят инвентаризацию компонент, входящих в состав ПО и формируют каталог этих компонент с учетом того, из какой библиотеки заимствована компонента, на каком языке написана и с какими другими связана. Эта информация предназначена прежде всего для руководителей разработки ПО в каталоге/перечне, который называется SBoM (Software Bill of Materials), содержатся также сведения руководителей разработки ПО, представителя заказчика и сотрудника информационной безопасности, контролирующего уровень безопасности. SBoM содержит метаданные о компонентах, сторонних сервисах, зависимостях, уязвимостях ПО. На основе этих данных можно спланировать изменения, связанные с устранениями уязвимостей, выявить наиболее подверженные атакам части, оценить уровень качества и безопасности ПО и его составных частей. Средства композиционного анализа формируют отчет о проведенном анализе в виде файла SARIF (Static Analysis Results Interchange Format), в котором содержатся результаты статического и композиционного анализа, которые позволяют оценить количество и уровень остаточных уязвимостей в ПО АС. Регулярное проведение обоих видов анализа обеспечивает возможность мониторинга/контроля качества и уровня безопасности, разрабатываемого АС.

Метод динамического анализа ПО АС основан на выполнении программы, однако, в отличие от функционального тестирования при динамическом анализе осуществляется не проверка выполнения функционального требования к тестируемой программе, а проверка наличия конкретных видов известных уязвимостей, систематизированных в классификациях OWASP, CWE, CVE, БДУ ФСТЭК. Тестирование серверной части клиент-серверных систем сопряжено с необходимостью анализа весьма большого количества маршрутов выполнения ПО АС. В этой связи применение метода динамического анализа оказывается не очень удобным в силу высокой трудоемкости ручных операций при

подготовке тестов и анализе результатов. С этой целью для тестирования безопасности серверного ПО чаще всего применяется фаззинг-тестирование.

Систематическое применение описанных средств и методов позволяет осуществлять мониторинг того, как устраняются уязвимости в ПО АС в ходе разработки и какие критические уязвимости могут попасть в ПО АС, передаваемого в постоянную эксплуатацию.

5. Заключение

Состав используемых методов тестирования зависит от размера ПО АС (количество строк кода) и нефункциональных требований к нему (наличие требований к режиму реального времени и требований по информационной безопасности).

1. Для небольших и средних по размеру комплексов ПО (несколько десятков тысяч строк), разрабатываемых по схеме DevOps, функциональное тестирование является основным видом тестирования и может проводиться поэтапно (в случае реализации ПО в виде нескольких микросервисов: каждый микросервис как отдельная сущность и интеграционное тестирование микросервисов на единой тестовой системе.
2. Для больших по размеру комплексов ПО (миллион и более строк кода), разработанных в виде набора микросервисов, которые разрабатываются каждый по схеме DevOps, а комплекс ПО интегрируется на отдельной инфраструктуре, определяемой потребностями заказчика и его требованиями к инфраструктуре эксплуатации АС. Тестовая система должна включать защищенные области (контура), доступ к которым осуществляется по политикам безопасности, разрабатываемым заказчиком для АС в целом [12].
3. Функциональное тестирование производится в основном для микросервисов, как объектов тестирования. При этом в рамках релизных циклов разработка микросервисов осуществляется по технологии DevSecOps. Повышается роль интеграционного тестирования, а системное тестирование становится обязательным. При этом объекты интеграционного и системного тестирования становятся также объектами тестирования безопасности, а проведение выявления и устранения уязвимостей на регулярной основе становятся обязательными перед вводом ПО АС в эксплуатацию.
4. Содержанием тестирования безопасности является использование статического и композиционного анализа в течение релизных циклов, а также при необходимости (для многозвенных клиент-серверных систем) дополнительно - использование средств динамического и фаззинг-тестирования. В настоящее время складывается технология использования динамического тестирования прежде всего для тестирования клиентской части ПО АС на регулярной основе, в том числе и в процессе эксплуатации системы – на комплексном стенде или резервной инфраструктуре параллельно с эксплуатацией ПО АС.
5. В организации, эксплуатирующей АС, как правило, устанавливают регламент использования системы для того, чтобы обеспечивать требования бизнеса по срокам сбора данных о функционировании организации, использующей АС, сроках формирования отчетности, выполнения временных требований по реализации бизнес-операций и т.п. Для оценки степени выполнения этого бизнес-регламента при текущем состоянии эксплуатационных характеристик АС, а также при перспективных требованиях бизнеса, оказывающих влияние на эксплуатационные характеристики АС, должно производиться нагрузочное тестирование на специализированном стенде или резервном сервере системы по программе и методике испытаний, специально разрабатываемой для каждого нагрузочного эксперимента. Такие эксперименты, как показывает опыт крупнейших

пользователей подобных корпоративных автоматизированных систем (банки, операторы связи, крупные энергетические компании и т.д.) целесообразно проводить периодически, но на плановой, постоянной основе. В этом случае должна быть разработана тестовая система для согласованных с заказчиком сценариев функционирования, критичных с точки зрения оцениваемых (паспортизуемых) эксплуатационных и нефункциональных характеристик АС. При этом должна быть разработана тестовая система для НТ, в которой учтены требования к чувствительным данным.

Список литературы / References

[1]. ГОСТ 34.602 – 2020. Информационные технологии/ КОМПЛЕКС СТАНДАРТОВ НА АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ Техническое задание на создание автоматизированной системы. М.: 9 с.

[2]. Батоврин В. К., Позин Б. А. Инженерия требований на современном промышленном предприятии // Программная инженерия. 2019. Т. 10. № 3. С. 114-124.

[3]. Липаев В.В. Тестирование крупных комплексов программ на соответствие требованиям, М.ИПЦ «Глобус», 2008, 376 с.

[4]. ГОСТ 34.603. Информационные технологии. Виды испытаний автоматизированных систем, 1992.

[5]. ГОСТ Р 71207-2024. Защита информации. Разработка безопасного программного обеспечения. Статический анализ программного обеспечения. Общие требования.

[6]. Позин Б.А. Ввод в действие информационных систем и сопровождение их программного обеспечения. - М.: Новые технологии, 2010. - 32 с. - (Прил. к журн. "Информационные технологии"; N 4/2010)

[7]. B. Pozin, I. Galakhov, Experience in automated functional and load testing in the life cycle of the mission-critical system. Baltic J. Modern Computing, Vol. 8 (2020), No. 2, 241-258, DOI: 10.22364/bjmc.2020.8.2.03.

[8]. Иванников В.П., Белеванцев А.А., Бородин А.Е., Игнатьев В.Н., Журихин Д.М., Аветисян А.И., Леонов М.И. Статический анализатор Svace для поиска дефектов в исходном коде программ. Труды Института системного программирования РАН. 2014;26(1): с.231-250.

[9]. Белеванцев А., Аветисян А. Многоуровневый статический анализ для поиска закономерностей ошибок и дефектов в исходном коде. В: Петренко А., Воронков А. (ред.) Перспективы системной информатики. PSI 2017. Конспекты лекций по информатике, том 10742, стр. 28–42.

[10]. Белеванцев А. (ИСП РАН) для форума "Russia DevOps Report - 2023”. [Электронный ресурс]. – 2023 – URL: https://russiadevopsreport.ru/. – (Дата обращения: 13.10.2024).

[11]. Позин Б.А., Бородушкина П.А., Коротков Д.А., Федоров М.А., Муратов А.Ф. Методика поиска уязвимостей в ПО, написанном на нескольких языках программирования. Труды ИСП РАН, том 37, вып. 1, 2025 г., стр. 121–132. DOI: 10.15514/ISPRAS–2025–37(1)–7.

[12]. Позин Б.А. Автоматизация и экономика для обеспечения жизненного цикла безопасного ПО, Информационная безопасность, № 4, 2024, с. 60.

[13]. Кулямин В.В. Обзор методов динамического анализа программного обеспечения. Труды Института системного программирования РАН. 2023; 35(4):7-44. DOI: 10.15514/ISPRAS-2023-35(4)-1.

Информация об авторах / Information about authors

Борис Аронович ПОЗИН – доктор технических наук, профессор, главный научный сотрудник ИСП РАН, профессор базовой кафедры ЗАО ЕС-лизинг в МИЭМ НИУ ВШЭ, технический директор ЗАО ЕС-лизинг. Сфера научных интересов: программная инженерия, системы обеспечения жизненного цикла доверенного программного обеспечения, автоматизированное тестирование программ.

Boris Aronovich POZIN – Dr. Sci. (Tech.), Professor, Chief Researcher at the ISP RAS, Professor of the Basic Department of CJSC EC-Leasing at the Higher School of Economics, Technical Director of CJSC EC-Leasing. Research interests: software engineering, life cycle ensuring systems for trusted software, automated software testing.