ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004

МЕТОД КОМПЛЕКСНОЙ ОЦЕНКИ ТЕСТОВОГО ПОКРЫТИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В НЕФТЕГАЗОВОМ СЕКТОРЕ (НА ПРИМЕРЕ ПРОЕКТОВ В AZURE DEVOPS)

Полетавкина Александра Сергеевна,
Уфимский государственный нефтяной технический университет
Уфа, Россия
e-mail: poletavkina162000@mail.ru

Аннотация

В статье рассматривается проблема обеспечения качества программного обеспечения в нефтегазовом секторе России в условиях импортозамещения. Обосновывается необходимость разработки собственных методов оценки тестового покрытия для минимизации рисков сбоев.

Проведен анализ существующих проблем: отсутствие прозрачности, недостаток метрик и неравномерность покрытия. Рассмотрены доступные инструменты, выявлены их ограничения, включая невозможность использования сторонних AI-решений в корпоративном периметре.

Предлагается метод комплексной оценки тестового покрытия для проектов в Azure DevOps, основанный на автоматизированном анализе покрытия API автоматизированными и ручными тестами. Представлен алгоритм реализации метода, включающий выгрузку данных из Swagger, анализ тестовых случаев и расчет процента покрытия.

Определены ограничения метода, связанные с отсутствием оценки модульных тестов из-за разобщенности контуров разработки и тестирования. Практическая значимость работы заключается в снижении временных и финансовых затрат, повышении надежности бизнес-процессов и формировании культуры принятия решений на основе объективных данных о качестве программного обеспечения.

Ключевые слова: тестовое покрытие, регрессионный тест план, Swagger, Azure DevOps, оценка качества программного обеспечения, автоматизация тестирования, нефтегазовая отрасль, управление дефектами

A METHOD FOR COMPREHENSIVE ASSESSMENT OF SOFTWARE TEST COVERAGE IN THE OIL AND GAS SECTOR (A CASE STUDY OF PROJECTS IN AZURE DEVOPS)

Poletavkina Aleksandra Sergeevna,
Ufa State Petroleum Technological University, Ufa, Russia
e-mail: poletavkina162000@mail.ru

ABSTRACT

The article addresses the problem of ensuring software quality in the Russian oil and gas sector under import substitution conditions. The necessity of developing proprietary test coverage assessment methods to minimize failure risks is substantiated.

An analysis of existing problems is carried out: lack of transparency, insufficient metrics, and uneven coverage. Available tools are reviewed, and their limitations are identified, including the impossibility of using third-party AI solutions within the corporate perimeter.

A method for comprehensive test coverage assessment for projects in Azure DevOps is proposed, based on automated analysis of API coverage by automated and manual tests. An algorithm for implementing the method is presented, including data extraction from Swagger, test case analysis, and coverage percentage calculation.

Limitations of the method are identified, related to the lack of unit test assessment due to the separation of development and testing environments. The practical significance of the work lies in reducing time and financial costs, increasing the reliability of business processes, and fostering a data-driven decision-making culture based on objective software quality data.

Keywords: test coverage, regression test plan, Swagger, Azure DevOps, software quality assessment, test automation, oil and gas, defect management

В условиях текущих геополитических вызовов и санкционного давления, приведших к оттоку зарубежного программного обеспечения и необходимости перехода на менее зрелые отечественные аналоги, перед нефтегазовым сектором России остро встала задача ускоренной цифровизации [1]. Для сохранения конкурентоспособности компаниям стратегически необходимо развивать собственные цифровые решения [2]. Однако ключевой проблемой в этой гонке становится обеспечение высокого качества сложного программного обеспечения, сбои в котором могут привести к миллионным убыткам и серьезным экологическим рискам.

Основной проблемой в оценке тестового покрытия является отсутствие унифицированной методики , которая позволяла бы объективно оценивать полноту тестового покрытия в рамках используемого в компании технологического стека [3].

При проведении тестирования тестировщики сталкиваются со следующими проблемами:

1. Отсутствие прозрачности: Тестировщики не имеют полной информации об охвате кода и требований из-за отсутствия или игнорирования отчетов о покрытии.
2. Нехватка измеримых метрик: нет единой системы метрик для оценки эффективности тестов, что не позволяет управлять качеством на основе объективных данных [4].
3. Неравномерное покрытие: разные модули продуктов имеют значительный разброс по уровню покрытия тестами [5].

Следствием этого является невозможность определить реальную эффективность автоматизированных тестов и оптимально распределить ресурсы. В результате критические функциональные требования могут иметь недостаточный уровень покрытия, что в условиях сжатых сроков выпуска релизов напрямую угрожает стабильности работы критически важных процессов.

Существующие на рынке инструменты трассировки требований и анализа покрытия кода являются либо зарубежными, либо разрозненными, не предоставляя целостной картины. Это подчеркивает необходимость разработки собственных решений.

Таким образом, совершенствование процессов оценки тестового покрытия является не просто внутренней задачей, а стратегическим императивом для обеспечения

технологического суверенитета и надежности цифровых активов компании в новых экономических реалиях.

Целью моего исследования является разработка метода комплексной оценки тестового покрытия для проектов, документация которых ведется в среде Azure DevOps [6]. Данный метод позволит автоматизировать анализ тестового покрытия API ручек автоматизированными тестами и ручными проверками в тестовых случаях.

При ведении разработки и тестирования в среде Azure DevOps существует ручной метод оценки, который заключается в проставлении связей между элементами.

Однако данный метод не учитывает наличие автоматизированных тестов, Главным минусом данного метода является тот факт, что оценить тестовое покрытие можно только функциональных требований ручными тест кейсами

Существует также ряд решений, которые позволяют оценить уровень тестового покрытия проекта, разрабатываемого как надстройки для Azure DevOps:

1. Использование Power BI и написание собственных скриптов для создания отчетов. Power BI позволяет выгружать данные из Azure DevOps и создавать на основе этих данных отчеты – с использованием кода и базового функционала. Аналогично можно делать запросы к Azure DevOps и с помощью программного кода формировать отчеты. Мой метод будет основан на этом принципе. Power BI был разработан компанией Microsoft, поэтому не поддерживается в периметре компании.
2. Инструменты для трекинга требований и связей (Modern Requirements). Система вычисляет процент требований, имеющих хотя бы один связанный тест-кейс. В основе данного метода также лежит составление матрицы трассировки. Данные инструмент не учитывает наличие автоматизированных тестов и считает только процент покрытия функциональных требований, а не конкретных API ручек. Более продвинутые решения, такие как Synapse RT, предлагают многофакторный анализ покрытия с использованием искусственного интеллекта для выявления скрытых зависимостей и оценки рисков. Тем не менее, использование сторонних AI-агентов неприемлемо в периметре многих крупных компаний из-за строгих требований к безопасности данных.

Предлагаемый метод оценки тестового покрытия основан на использовании Swagger с перечнем API ручек проекта, тестовых случаев в Azure DevOps, проекта автоматизированных тестов и скрипта для выгрузки, анализа данных и формирования отчета [7]. Подключение к проекту будет происходить с использованием токена. Формирование токена необходимо для получения доступа к проекту и осуществляется в настройках Azure DevOps любым пользователем, который имеет права на чтение данного проекта. Ввод токена и ссылки на проект будет первичной настройкой для использования разрабатываемого решения.

Алгоритм оценки уровня покрытия будет включать следующие шаги:

1. Подключение к Swagger и выгрузка всех API ручек
2. Поиск полученных ручек в проекте автоматизированных тестов
3. Расчет процента покрытия API ручек автоматизированными тестами как отношение количества покрытых ручек к общему числу ручек из Swagger.
4. Выполнение запроса к Azure DevOps для получения ID всех тестовых случаев, входящих в регрессионный тестовый план.
5. Выполнение запроса к Azure DevOps для получения описания шагов тестовых случаев по ID, полученных в предыдущем запросе
6. Выделение из описания шагов перечня затрагиваемых API ручек
7. Расчет процента покрытия API ручек ручными тестами как отношение количества покрытых ручек к общему числу ручек из Swagger.

8. Вывод результатов

Ограничениями предложенного метода будет являться отсутствие оценки покрытия кода модульными тестами. Причиной этому являются следующие факторы:

1. Постановки задач, тест кейсы, автоматизированные тест находятся в Azure DevOps, который находится в корпоративной сети передачи данных. Модульные тесты пишутся разработчиками в DevZone. Это закрытых контура для ведения работ, которые не имеют интеграции между собой. Единственным способом переносить юнит тесты из одного контура в другой является ручная пересылка через корпоративную почту. Однако данный способ чреват нарушения безопасности и неэффективность. Модульные тесты пишутся регулярно, а значит проводить актуализацию репозитория модульных тестов в корпоративной сети передачи данных тоже придется регулярно.
2. При написании Unit тестов разработчики не помечают к привязке к конкретной карточке с техническими требованиями или API ручкам, поэтому провести трассировку между юнит тестом и конкретным функциональным требованием нет возможности на данный момент. Есть возможность лишь с помощью отчета о покрытии отследить уровень покрытия различных сервисов. Но эти цифры не имеют ценности для тестировщиков, поскольку не имеют привязку к конкретному функционалу, а значит и повлиять на этот показатель у тестировщиков нет возможности.

Научно-методические нововведения представляют собой:

1. Уникальная формула оценки тестового покрытия как со стороны автоматизированных тестов, так и со стороны ручных тестов
2. Адаптация под ограничения компании: локальное решение без внешних AI-сервисов с возможностью дальнейшего расширения метрик и применения локального AI агента для тех проектов, которые находятся в DevZone.
3. Практическая ориентированность: учет корпоративных регламентов и реальных процессов ведения разработки и тестирования компании.

Таким образом, предлагаемое решение представляет собой целостный подход к управлению качеством программного обеспечения, который позволяет системно решить проблему оценки тестового покрытия.

Конкретными практическими результатами внедрения станут снижение расходов за счет сокращения времени на поиск причин ошибок и устранения дублирующих проверок. Это напрямую повысит надежность критически важных бизнес-процессов благодаря гарантированному охвату тестами функциональности и непрерывному мониторингу качества.

Решение способствует формированию культуры принятия решений на основе данных, предоставляя всем участникам проекта единую систему показателей, наглядное отображение динамики ключевых параметров и автоматизированные сводки о готовности к выпуску новой версии.

В конечном счете, внедрение данного подхода позволит не только решить текущие задачи, но и создать прочное преимущество компании за счет выстроенной эталонной системы управления качеством, что полностью соответствует стратегическим целям цифрового развития и укрепляет позиции организации как технологического лидера в своей области.

Список литературы:

1. Абдуллин А.У. Социально-экономическое развитие России в условиях санкций: риски и механизмы адаптации // Экономика и управление: научно-практический журнал. 2024. № 2. С. 24–30.

2. Указ президента РФ от 07.05.2024 № 309 «О национальных целях развития Российской Федерации на период до 2030 года и на перспективу до 2036 года» [Электронный ресурс] // Официальное опубликование правовых актов. URL: http://publication.pravo.gov.ru/document/0001202405070015 (дата обращения 17.01.2026)
3. Куликов, С. С. Тестирование программного обеспечения. Базовый курс. Минск: Четыре четверти, 2020. 312 с.
4. Проскуряков А. В. Качество и тестирование программного обеспечения. Метрология программного обеспечения: учебное пособие. Ростов-на-Дону: Издательство Южного федерального университета, 2022. 197 с.
5. Врагова, А. С. Построение модели классов для формирования матрицы трассировки требований к программному обеспечению // Инжиниринг предприятий и управление знаниями (ИП&УЗ-2018): сборник научных трудов XXI Российской научной конференции. 2018. № 2. С. 152-156.
6. Hewage M. A Practical Guide to Azure DevOps. USA: Independently published, 2019. 170 p.
7. Буйначев С. К. Основы программирования на языке Python: учебное пособие. Екатеринбург: Изд-во Урал. ун-та. 2014. 91 с.

References:

1. Abdullin A.U. Socio-economic development of Russia under sanctions: risks and adaptation mechanisms // Economics and Management: Scientific and Practical Journal. 2024. No. 2. Pp. 24–30.
2. Decree of the President of the Russian Federation dated 05/07/2024 No. 309 "On the National Development Goals of the Russian Federation for the Period up to 2030 and for the Future up to 2036" [Electronic resource] // Official Publication of Legal Acts. URL: http://publication.pravo.gov.ru/document/0001202405070015 (accessed: 17.01.2026)
3. Kulikov S.S. Software Testing. Basic Course. Minsk: Chetyre chetverti, 2020. 312 p.
4. Proskuryakov A.V. Software Quality and Testing. Software Metrology: textbook. Rostov-on-Don: Southern Federal University Publishing House, 2022. 197 p.
5. Vragova A.S. Constructing a Class Model for Forming a Software Requirements Traceability Matrix // Enterprise Engineering and Knowledge Management (IE&KM-2018): collection of scientific papers of the XXI Russian Scientific Conference. 2018. No. 2. Pp. 152-156.
6. Hewage M. A Practical Guide to Azure DevOps. USA: Independently published, 2019. 170 p.
7. Buinachev S.K. Fundamentals of Programming in Python: textbook. Yekaterinburg : Ural University Publishing House. 2014. 91 p.