РАЗРАБОТКА ЧЕК-ЛИСТОВ И ПРИЕМОЧНЫХ КРИТЕРИЕВ ДЛЯ ТИПОВЫХ И УНИКАЛЬНЫХ DRUPAL-МОДУЛЕЙ

Кулецкая Елена Александровна,
старший инженер по обеспечению качества, специалист по обеспечению качества в области корпоративных веб-систем и систем на базе-Drupal, Крикетт хилл роуд, Вэстовер, штат Мэрилэнд

E-mail: e.kuletskaia@gmail.com

Аннотация. Статья посвящена разработке и обоснованию подхода к формированию чек-листов и приемочных критериев для модулей Drupal, включая типовые модули сообщества и уникальные модули, создаваемые под специфические требования проекта. Рассматриваются теоретические основы качества модульной разработки в Drupal: модульная архитектура, управление зависимостями через Composer, сервисная модель и внедрение зависимостей, а также уровни автоматизированного тестирования. Предложена структура чек-листа приемки типового модуля и набор критериев приемки для уникальных модулей с акцентом на конфигурационную модель Drupal (Configuration API, схема конфигурации), обновляемость и подтверждаемость требований через артефакты разработки. Практическая апробация методики описывается как интеграция критериев приемки в процесс CI/CD с фиксацией результатов пайплайна, отчетов тестирования и процедур управления конфигурацией.

Ключевые слова: Drupal, модули, качество, чек-лист, приемка, custom-модуль, Configuration API, Composer, тестирование, CI/CD, безопасность.

Актуальность исследования

Актуальность исследования обусловлена объективными требованиями к обеспечению качества, безопасности и устойчивости современных веб-систем. Drupal является одной из распространённых систем управления контентом с модульной архитектурой, в рамках которой функциональность формируется за счёт сочетания ядра, модулей сообщества и индивидуально разработанных модулей. Такая модель повышает гибкость платформы, однако одновременно увеличивает риски, связанные с неоднородностью кода, различиями в архитектурных подходах и уровне проработки требований.

Практика разработки показывает, что в проектах нередко отсутствуют формализованные процедуры приемки модулей. Проверка качества часто ограничивается функциональным тестированием без системной оценки архитектурной корректности, соответствия стандартам кодирования, требований безопасности и производительности. В результате возрастает вероятность накопления технического долга, появления уязвимостей и усложнения

дальнейшего сопровождения системы. Для платформы с развитой экосистемой модулей данный фактор приобретает особое значение.

Дополнительную актуальность теме придаёт рост требований к информационной безопасности веб-приложений. Регулярная публикация бюллетеней безопасности и обновлений для модулей свидетельствует о необходимости предварительной оценки рисков перед внедрением сторонних решений. Формализация критериев приемки позволяет систематизировать проверки, снизить вероятность использования уязвимого или архитектурно некорректного кода и обеспечить воспроизводимость процесса контроля качества.

Кроме того, развитие DevOps-подходов и автоматизированных процессов CI/CD требует перевода качественных требований в измеряемые и проверяемые показатели. Чек-листы и приемочные критерии создают основу для автоматизации контроля: их можно интегрировать в процедуры code review, статического анализа, тестирования и релизного контроля. Таким образом, разработка методически обоснованных чек-листов для типовых и уникальных Drupal-модулей является актуальной научно-практической задачей, направленной на повышение управляемости, надежности и сопровождаемости программных решений.

Цель исследования

Целью данного исследования является обоснование и разработка методически согласованной системы чек-листов и приемочных критериев для типовых и уникальных Drupal-модулей, обеспечивающей воспроизводимую проверку качества, безопасности, обновляемости и сопровождаемости модульных решений, а также возможность внедрения этих критериев в автоматизированные процессы разработки и поставки.

Материалы и методы исследования

Материалами исследования выступили открытые источники и официальная документация по Drupal (архитектурные подходы, Configuration API, принципы тестирования и обновления модулей), а также общедоступные отраслевые рекомендации по безопасной разработке и практики DevOps, применяемые при автоматизации контроля качества.

В работе использованы методы анализа и обобщения нормативно-методических источников, сравнительный анализ подходов к приемке типовых и уникальных модулей, структурирование требований в виде проверяемых критериев, а также метод практической апробации через включение критериев приемки в процесс CI/CD с фиксацией результатов пайплайна и тестов.

Результаты исследования

Качество модульной разработки в Drupal целесообразно рассматривать не как «набор вкусовых правил», а как совокупность проверяемых требований к совместимости, расширяемости и предсказуемости поведения кода в экосистеме, где одно и то же ядро обслуживает разные сайты и команды. Архитектурная основа Drupal опирается на четко определённые механизмы расширения (модули подключают функциональность, взаимодействуют через API и точки

расширения) и на разделение ответственности между слоями: хранение данных, API сущностей, инфраструктурные сервисы, а также слой представления. Эту идею модульности и взаимосвязи компонентов на практике иллюстрируют схемы взаимодействия модулей и подсистем (рисунок 1).

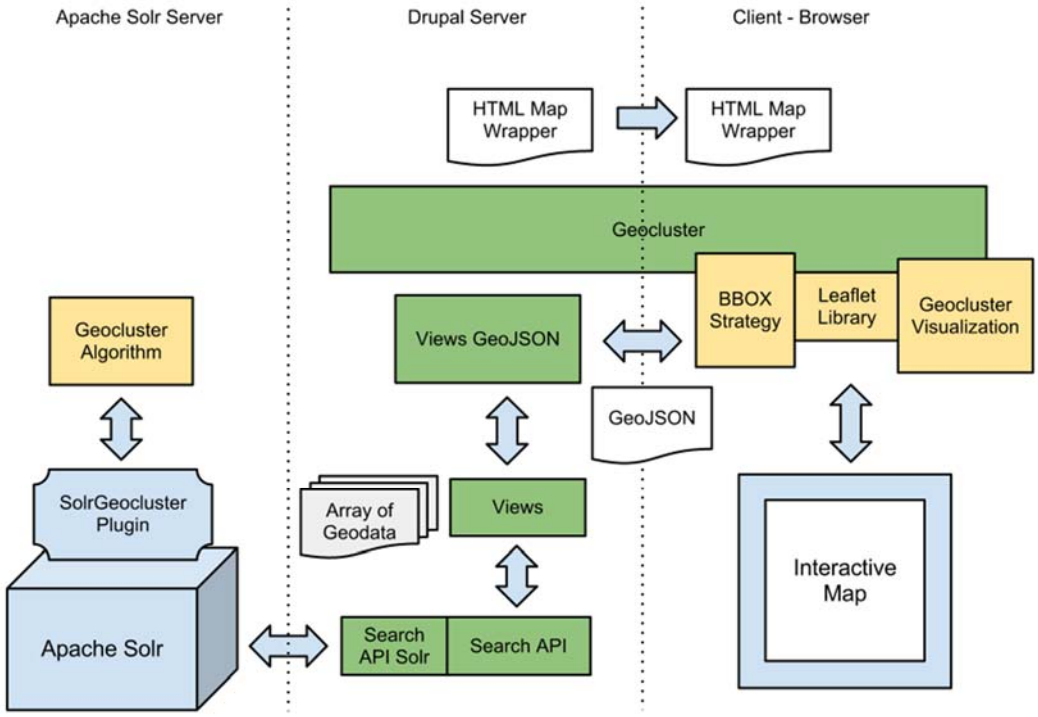

Рис. 1 Архитектурная схема интеграции геокластеризации в Drupal с использованием Apache Solr, Search API и библиотеки Leaflet [3]

Существенной теоретической опорой качества в Drupal является переход платформы к современной PHP-инженерии: управление зависимостями через Composer, активное использование внешних библиотек и компонентов, а также формализация состава проекта через composer.json. В результате качество модуля определяется не только тем, «работает ли функциональность», но и тем, корректно ли описаны зависимости, отсутствуют ли конфликтующие версии пакетов, воспроизводима ли сборка на разных средах. Drupal прямо фиксирует, что ядро использует Composer для управления собственными зависимостями (в том числе компонентами Symfony и HTTP-клиентскими библиотеками), а также допускает использование Composer для управления зависимостями модулей и сборки сайта в целом [5].

Понимание качества модульной разработки невозможно без учета жизненного цикла ядра и практик релизного управления, поскольку приемочные критерии модуля должны соответствовать целевой версии Drupal и её изменениям. На сегодняшний день ключевыми вехами стали выход Drupal 9 (как продолжение APIs Drupal 8.9) и выход Drupal 10, а также подготовка Drupal 11 в

2024 году. Drupal 9.0.0 был выпущен 3 июня 2020 года одновременно с Drupal 8.9.0, при этом отмечалось, что обе ветки имеют одинаковые API и функциональность, что концептуально поддерживает идею совместимости модулей между близкими версиями [2].

Drupal 10.0.0 был выпущен в декабре 2022 года и позиционировался как первый поддерживаемый релиз новой мажорной версии, при этом для более плавного обновления рекомендовалось сначала перейти на Drupal 9.5.

Отдельный пласт теоретических основ качества в Drupal – сервисная архитектура и внедрение зависимостей. Drupal использует контейнер сервисов поверх контейнера Symfony: сервисы определяются и предоставляются модулем или ядром, а зависимости разрешаются контейнером, что позволяет унифицировать доступ к инфраструктуре (например, к базе данных, логированию, переводу интерфейса). Это напрямую связано с приемочными критериями: предпочтение отдаётся внедрению зависимостей в классы вместо «жестких» вызовов, потому что так код проще тестировать, сопровождать и расширять без побочных эффектов.

Качество модулей также опирается на формализацию уровня тестирования. В Drupal в качестве базовых тестовых опорных точек используются специализированные базовые классы, которые задают «профиль» теста. На уровне теории это соответствует идее многоуровневого тестирования (обобщенно – «пирамида тестов», рисунок 2), но в Drupal важно именно то, что каждый уровень тестов имеет свой контракт: пространство имен, расположение файлов, степень зависимости от инфраструктуры.

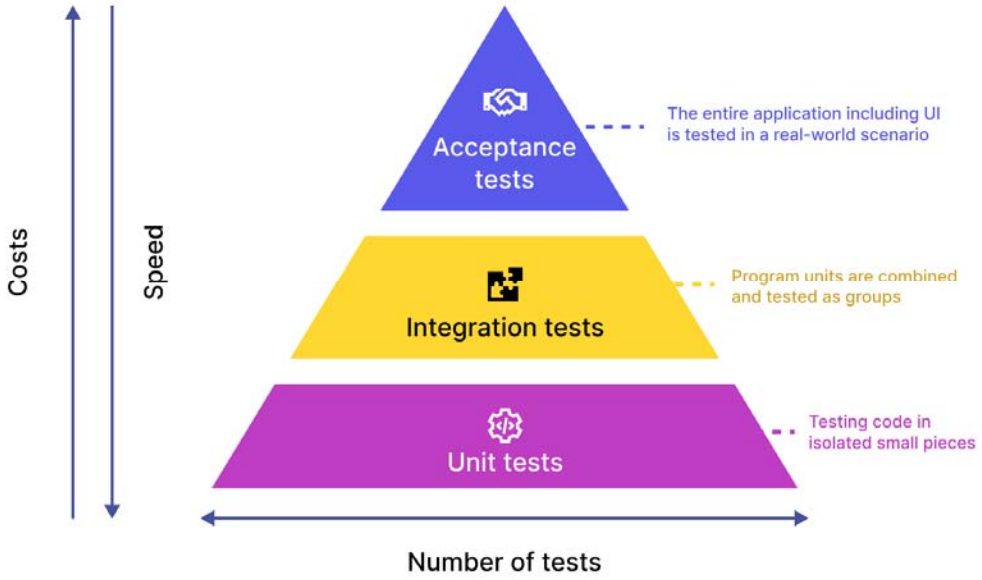

Рис. 2 Пирамида тестирования программного обеспечения [4]

Уровни автоматизированного тестирования модулей в Drupal представлены в таблице 1.

Таблица 1

Уровни автоматизированного тестирования модулей в Drupal

Уровень тестирования	Базовый класс (ядро Drupal)	Что проверяется
Модульные тесты	Drupal\Tests\UnitTestCase	Логика отдельных классов и методов при минимальном количестве зависимостей
Ядерные (интеграционные) тесты	Drupal\KernelTests\ KernelTestBase	Взаимодействие модуля с подсистемами ядра при инициализации тестового ядра Drupal
Функциональные тесты	Drupal\Tests\BrowserTestBase	Поведение системы при HTTP-взаимодействии с имитацией работы браузера
Функциональные тесты с поддержкой JavaScript	Drupal\FunctionalJavascriptTests \WebDriverTestBase	Поведение пользовательского интерфейса в реальном браузере с поддержкой JavaScript

Источник: [1].

Практически чек-лист должен фиксировать минимально необходимое «ядро» требований: соблюдение стандартов кодирования, наличие достаточной документации, учет базовых требований безопасности и управляемость обновлений (включая реакцию на публикации об уязвимостях и выпуск исправлений). Критично, чтобы у каждого пункта была форма подтверждения: что именно проверяем и какой результат считается выполнением (таблица 2).

Таблица 2

Минимальная структура чек-листа приемки типового модуля и способы подтверждения выполнения требований

Раздел чек-листа	Критерий выполнения	Подтверждение при приемке
Стандарты кода	Программный код соответствует стандартам Drupal и общепринятым стандартам разработки на PHP	Результаты проверки линтером и средствами статического анализа; отсутствие существенных замечаний по итогам проверки кода
Документация	Документация обеспечивает возможность установки, настройки и сопровождения модуля	Наличие файла README или описания проекта; указание зависимостей; описание порядка установки и удаления; сведения о конфигурации и ограничениях; описание прав доступа при их наличии
Безопасность	Реализованы базовые требования безопасной разработки	Проверка по перечням требований безопасной разработки (например, OWASP ASVS и OWASP Top 10) в части контроля доступа, проверки входных данных, обработки ошибок и ведения журналов
Управление уязвимостями и обновлениями	Обеспечен контроль обновлений и снижение рисков эксплуатации	Наличие порядка отслеживания уведомлений о безопасности и новых релизов; установленная процедура обновления и тестирования перед внедрением; ведение журнала изменений

Источник: разработка автора.

Для уникальных модулей, которые создаются под конкретные бизнес-процессы и интеграции проекта, критерии приемки должны опираться на проверяемые свойства, закреплённые в документации Drupal: корректную работу с конфигурацией, предсказуемость обновлений и наличие автоматизированной проверки ключевой логики. Drupal предусматривает использование схемы (schema/metadata) для YAML-конфигурации: она задаёт типы и метаданные и применяется как часть конфигурационной модели платформы. Поэтому в приемке custom-модуля отдельно фиксируется наличие и корректность схемы для собственных настроек (а не только наличие самих YAML-файлов), поскольку это напрямую связано с корректностью интерпретации структуры конфигурации. Если модуль хранит «простые» настройки, их следует получать и изменять через Simple Configuration API, который отдельно описан как механизм работы с простой конфигурацией (и прямо отделён от конфигурационных сущностей).

При разработке критериев важно различать простую конфигурацию и конфигурационные сущности: в документации Drupal подчёркивается, что конфигурационные сущности предназначены для списков настраиваемых пользователем объектов (например, представления, стили изображений) и сопровождаются набором CRUD-хуков, как у обычных сущностей (таблица 3). Это различие используется как правило приемки: если настройки предполагают множественные экземпляры, создание и удаление пользователем и реакцию других модулей, то приемка должна подтверждать, что реализован именно подход конфигурационных сущностей, а не «сложная простая конфигурация».

Таблица 3

Минимальный набор критериев приемки
для уникального Drupal-модуля и проверяемые артефакты

Область приемки	Что подлежит проверке	Подтверждение при приемке
Конфигурация и схема	Наличие и корректность схемы конфигурации; согласованность структуры настроек	Наличие файла schema/metadata для конфигурации; отсутствие ошибок при работе с конфигурацией; соответствие установленному формату, описанному в документации Drupal
Работа с настройками	Использование Configuration API для чтения и записи параметров	Проверка сценариев изменения и сохранения настроек; соответствие требованиям Simple Configuration API
Обновления и совместимость	Корректность обновления конфигурационных сущностей при внесении изменений	Наличие процедур обновления; успешное выполнение сценария обновления в соответствии с официальной документацией по обновлению конфигурационных сущностей
Автоматизированное тестирование	Наличие тестов для ключевой логики и интеграционных сценариев	Отчёт о выполнении тестов PHPUnit; применение соответствующих базовых классов тестирования (в том числе уровня Kernel при необходимости взаимодействия с подсистемами ядра)
Учет требований безопасности	Учет рисков при разработке и эксплуатации модуля	Соответствие рекомендациям сообщества по безопасной разработке; учет процедур реагирования на уязвимости и требований Drupal Security Team

Источник: разработка автора.

Практическая апробация методики приемки оформляется как внедрение чек-листов и критериев в воспроизводимый процесс поставки, где итог приемки подтверждается сохраняемыми результатами сборки и тестирования. В терминах CI/CD это означает, что критерии приемки «привязываются» к автоматизированным проверкам и выполняются одинаково при каждом изменении кода. Связка с CI/CD обычно реализуется через пайплайн.

Для Drupal-модулей важной частью практической приемки является проверка переносимости конфигурации между окружениями. Drupal описывает рабочий процесс переноса конфигурации с использованием Drush: экспорт выполняется командой drush config:export (также используется сокращение drush cex), при этом конфигурация выгружается в каталог синхронизации. В апробации методики это превращается в конкретное подтверждение: конфигурация модуля экспортируется и затем импортируется на целевом окружении без ошибок, что доказывает воспроизводимость настроек.

Отдельно проверяется корректность обновлений модуля. Официальная документация Drupal API указывает, что для изменений, требующих действий при обновлении сайта, в модуле следует добавлять реализации hook_update_N(), которые вызываются в процессе обновления. В апробации это означает, что обновление версии custom-модуля на тестовом стенде должно проходить без ошибок, а все необходимые обновляющие процедуры выполняются штатно.

Наконец, приемка подтверждается запуском автоматизированных тестов. Drupal прямо рекомендует писать новые тесты, используя базовые классы PHPUnit, включая KernelTestBase для случаев, когда требуется частичная загрузка ядра и проверка интеграции с подсистемами Drupal. По итогам апробации сохраняются артефакты: статус пайплайна, отчеты о тестах и результаты операций экспорта/импорта конфигурации, что позволяет документально подтвердить выполнение критериев приемки.

Выводы

Разработка чек-листов и приемочных критериев для Drupal-модулей позволяет перевести оценку качества из субъективной плоскости в формат проверяемых требований, подтверждаемых артефактами разработки и тестирования. Для типовых модулей обоснована минимальная структура приемки, включающая стандарты кода, документацию, безопасность и управляемость обновлений. Для уникальных модулей показана необходимость расширения критериев за счет проверки конфигурационной модели (наличие и корректность схемы конфигурации, корректное использование Configuration API), обновляемости (включая обновляющие процедуры) и обязательной автоматизированной проверки ключевой логики.

Практическая апробация демонстрирует, что интеграция критериев приемки в CI/CD и процедуры управления конфигурацией повышает воспроизводимость приемки и снижает риски эксплуатации за счет прозрачной фиксации результатов проверок.

Литература:

1. Class UnitTestCase [Электронный ресурс]. – Режим доступа: https://api.drupal.org/api/drupal/core%21tests%21Drupal%21Tests%21UnitTestCase.php/class/UnitTestCase/11.x.
2. Drupal 9.0.0 [Электронный ресурс]. – Режим доступа: https://www.drupal.org/project/drupal/releases/9.0.0.
3. Drupal Mapping Diagrams [Электронный ресурс]. – Режим доступа: https://www.drupal.org/node/1807358.
4. Test Pyramid Explained: Strategy, Levels & Examples [Электронный ресурс]. – Режим доступа: https://katalon.com/resources-center/blog/test-pyramid.
5. Using Composer with Drupal [Электронный ресурс]. – Режим доступа: https://www.drupal.org/docs/develop/using-composer/using-composer-with-drupal.