Интеграционное тестирование

Автоматизированное тестирование можно проводить на следующих уровнях:

• модульное тестирование;

• интеграционное тестирование;

• функциональное тестирование;

• тестирование интерфейса пользователя;

• сквозное тестирование.

Интеграционное тестирование предполагает тестирование части системы. Главная задача тестирования на этом уровне — это поиск ошибок, связанных с неверным взаимодействием между отдельными модулями. При интеграционном тестировании так же необходимы заглушки для замены отсутствующих в данный момент модулей. Интеграционное "тестирование может иметь разный масштаб, который зависит от количества вовлекаемых в тестирование модулей. При этом используется один из двух подходов

1. Монолитный подход — одновременное объединение всех модулей программы.

2. Инкрементный подход — постепенное добавление модулей с тестированием промежуточной версии после добавления каждого нового модуля

Щербак, А. В.

Тестирование программного обеспечения : учебник для вузов / А. В. Щербак. — Москва : Издательство Юрайт, 2025. — 145 с. — (Высшее образование). — ISBN 978-5-534-19291-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 49 — URL: https://urait.ru/bcode/580604/p.49 (дата обращения: 11.04.2025).

Модульное тестирование

Модульное тестирование подразумевает тестирование отдельных модулей с целью выявления ошибок, находящихся в этих самых модулях, и определение готовности к переводу на следующий уровень разработки. Такой подход предполагает наличие так называемых заглушек на всех интерфейсах модуля. Заглушки используются для ввода исходных значений, для выдачи результатов и т. д. На данном уровне тестирования могут быть обнаружены ошибки в алгоритме и в его переносе на конкретный язык программирования. В то же время ошибки, касающиеся интерфейсов, производительности и совместимости, на этом уровне пропускаются, поскольку здесь еще нет рассмотрения программы как чего-то целостного. Организации, накопившие достаточный опыт разработки ПО, имеют базу данных прежних разработок, на основе которой они могут принимать решения о расстановке акцентов на данном уровне тестирования."

Щербак, А. В.

Тестирование программного обеспечения : учебник для вузов / А. В. Щербак. — Москва : Издательство Юрайт, 2025. — 145 с. — (Высшее образование). — ISBN 978-5-534-19291-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 77 — URL: https://urait.ru/bcode/580604/p.77 (дата обращения: 11.04.2025).

Рефакторинг кода

Необходимым условием для непрерывного рефакторинга является наличие тестов, которые позволяют убедиться в том, что система продолжает работать, хотя внутри системы произошли изменения. Тесты на уровне модулей не могут дать такую возможность и сами в большой степени подвержены изменениям при рефакторинге — какие-то тесты нужно переработать, а какие-то просто убрать. Соответственно, если команда концентрируется на модульных тестах, то лишь малая ее часть следит за сохранением общего функционирования. И может стать сложной задачей определение того, какие тесты можно оставить, а какие должны быть пересмотрены из-за реструктурирования кода. Как результат — чрезвычайно медленное движение далее при сохранении фантастически высоких показателей покрытия тестами.

Щербак, А. В.

Поддержка и тестирование программных модулей : учебник для среднего профессионального образования / А. В. Щербак. — Москва : Издательство Юрайт, 2025. — 145 с. — (Профессиональное образование). — ISBN 978-5-534-19290-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 84 — URL: https://urait.ru/bcode/580603/p.84 (дата обращения: 11.04.2025).

Оптимизация кода

оптимизация кода. И деобфускация, и оптимизация программного кода в той или иной степени противоположны процессу обфускации. В процессе обфускации в программный код часто производится добавление лишних операций, которые обычно никак не влияют на результаты работы самой программы и предназначены для затруднения процесса изучения кода программы посторонними лицами. Следует отметить, что большинство компиляторов в процессе компиляции исходного кода автоматически осуществляют и процесс оптимизации, поэтому, если процесс обфускации осуществляется над исходным кодом программы (обфускация высокого уровня), возникает определенная вероятность того, что ее эффективность после процесса компиляции снизится. Если же такой исходный код будет подвержен обработке интерпретатором (т.е. не будет подвержен компиляции), эффективность выполненного процесса обфускации не изменится.

Казарин, О. В.

Надежность и безопасность программного обеспечения : учебник для вузов / О. В. Казарин, И. Б. Шубинский. — Москва : Издательство Юрайт, 2025. — 342 с. — (Высшее образование). — ISBN 978-5-534-05142-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. с. 221 — URL: https://urait.ru/bcode/563862/p.221 (дата обращения: 11.04.2025).