Программирование: как подготовиться к проверке кода ИИ-агентом в 2026 году

19.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В 2026 году ИИ-агенты пишут не только отдельные функции, но и целые модули, тесты и конфигурации. По данным опросов разработчиков, доля кода, созданного с помощью ИИ, в некоторых командах достигает 40–60%. При этом генеральный директор Microsoft Сатья Наделла в ноябре 2026 года подчеркнул: компания не собирается заменять инженеров-программистов нейросетями, а перестраивается так, чтобы каждый сотрудник использовал ИИ для работы. Это значит, что проверка кода, написанного агентом, становится такой же обязательной частью процесса, как и написание тестов.

Почему обычный code review не справляется

Code review — это процесс проверки изменений в коде другими разработчиками до слияния в основную ветку. Когда код пишет человек, ревьюер может опереться на контекст: зачем автор так сделал, какие были альтернативы, где он сомневался. Агент такого контекста не даёт: он выдаёт результат, который может выглядеть правдоподобно, но содержать логические дыры, устаревшие вызовы API или неявные допущения.

Кроме того, агент не устаёт и не отвлекается, поэтому может сгенерировать больше кода, чем команда успевает осмысленно проверить. Возникает риск «резинового ревью»: когда ревьюер просто просматривает diff, не вникая в детали.

Что проверять в первую очередь

Чтобы ревью кода от агента было эффективным, стоит разделить проверку на уровни. Вот минимальный чек-лист.

1. Контекст и цель

- Есть ли в задаче или pull request описание, зачем этот код нужен и какие у него границы?
- Соответствует ли решение архитектуре проекта или вводит новый паттерн без обоснования?
- Не дублирует ли код уже существующие функции?

2. Корректность и безопасность

- Обрабатываются ли ошибки и исключения? Агенты часто «забывают» про негативные сценарии.
- Нет ли утечек секретов, ключей или персональных данных в коде и логах?
- Проверены ли входные данные и граничные условия?
- Используются ли актуальные версии библиотек и API? В 2026 году многие фреймворки обновляются чаще, чем раз в квартал.

3. Тесты и воспроизводимость

- Покрыт ли новый код тестами? Если агент написал тесты, проверьте, что они действительно проверяют поведение, а не просто повторяют реализацию.

- Можно ли воспроизвести сборку и запустить тесты локально или в CI без ручных правок?

Как организовать процесс ревью

Просто добавить ещё одного ревьюера недостаточно. Нужны правила, которые учитывают особенности агентского кода.

1. **Ограничьте размер изменений.** Один pull request — одна логическая задача. Если агент сгенерировал 1000 строк, разбейте их на части. Большие diff'ы ревьюеры просматривают поверхностно.
2. **Требуйте «след» решения.** Пусть агент или разработчик, запустивший агента, прикладывает краткое пояснение: какие альтернативы рассматривались, какие допущения сделаны, какие места вызывают сомнения.
3. **Введите обязательную проверку безопасности.** Для кода, который работает с аутентификацией, платежами или персональными данными, добавьте отдельный шаг — статический анализ и, при необходимости, ручной аудит.
4. **Используйте автоматические линтеры и анализаторы.** Они снимают рутину: форматирование, неиспользуемые импорты, потенциальные уязвимости. В 2026 году популярные линтеры (ESLint, Ruff, golangci-lint) имеют правила, ориентированные на код, сгенерированный ИИ.
5. **Проверяйте тесты отдельно.** Тесты, написанные агентом, могут быть «зелёными», но бесполезными. Оцените, покрывают ли они граничные случаи и ошибки.

Метрики, которые помогают не потерять качество

Чтобы понять, не ухудшилось ли качество кода после внедрения агентов, стоит отслеживать несколько показателей. Они не заменяют экспертизу, но дают сигнал.

- **Доля кода, возвращённого на доработку после ревью.** Если она растёт, возможно, агенты генерируют слишком сложные или некачественные изменения.
- **Количество дефектов, найденных после релиза, в коде, написанном агентом.** Сравните с дефектами в коде, написанном людьми.
- **Время на ревью одного pull request.** Слишком быстрое ревью — признак формального подхода.
- **Покрытие тестами и мутационное тестирование.** Последнее показывает, насколько тесты чувствительны к изменениям в коде.

  Важно: метрики не должны превращаться в KPI, иначе разработчики начнут оптимизировать цифры, а не качество. Используйте их для обсуждения в команде.

Инструменты и автоматизация

В 2026 году на рынке есть инструменты, которые помогают ревьюить код, написанный ИИ. Например, статические анализаторы с поддержкой правил для популярных языков, сканеры секретов, системы проверки зависимостей. Однако не стоит полагаться только на автоматику: она не понимает бизнес-контекст.

Полезно настроить CI-пайплайн так, чтобы он блокировал слияние при обнаружении критичных проблем: уязвимостей, падения тестов, нарушения лицензий. Это снимает часть нагрузки с ревьюеров.

Организационные правила

Технические меры работают, только если есть договорённости в команде. Зафиксируйте их в документации проекта или в внутреннем гайде.

- Кто отвечает за код, сгенерированный агентом? Ответственность всегда на человеке, который запустил агента и создал pull request.
- Какие изменения требуют обязательного ревью двумя разработчиками? Например, код, работающий с деньгами или доступами.
- Как поступать, если ревьюер не уверен в решении? Лучше вернуть на доработку или обсудить в чате, чем пропустить сомнительный код.

Что в итоге

ИИ-агенты ускоряют написание кода, но не отменяют инженерную дисциплину. Проверка кода, написанного агентом, требует внимания к контексту, безопасности и тестам. Чек-листы, ограничение размера изменений, автоматические анализаторы и прозрачные правила помогают сохранить качество. Как показал пример Microsoft в ноябре 2026 года, компании перестраивают процессы вокруг ИИ, но не отказываются от инженеров. Значит, навык проверять код, созданный ИИ, становится базовым для разработчика.

Источник: https://cms.dolinais.ru/programmirovanie-kak-podgotovitsya-k-proverke-koda-ii-agentom-v-2026-godu