За пределами отчёта о статусе: использование LLM для выявления реального состояния разработки SaMD

Источник: MedTech Intelligence

Оригинал: https://medtechintelligence.com/feature_article/beyond-the-status-report-using-llms-to-reveal-the-true-state-of-samd-development/

LLM | SaMD | ИИ в разработке | медицинские устройства | разработка ПО | регуляторное соответствие | управление качеством

Панели проектов, базы данных требований, доски спринтов и отчёты о статусе пытаются ответить на этот вопрос. Однако в сложной разработке Программного обеспечения как медицинского изделия (Software as a Medical Device, SaMD) эти инструменты часто описывают состояние проекта, а не состояние продукта.

Требование может быть помечено как реализованное, даже если связанное с ним управление рисками не было полностью интегрировано. Пользовательская история программного обеспечения может быть закрыта, потому что основная логика приложения завершена, в то время как пользовательский рабочий процесс, обработка данных, управление исключениями или доказательства верификации остаются незавершёнными. И наоборот, функциональность может уже существовать в кодовой базе, но оставаться отключённой от требований, тестов или артефактов планирования.

Эти расхождения часто обнаруживаются во время системной интеграции, формальной верификации или обзоров дизайна — именно тогда, когда их исправление становится более дорогостоящим.

Большие языковые модели (Large Language Models, LLM), интегрированные в среды разработки, создают возможность сократить этот разрыв. Их наиболее важный вклад в регулируемую разработку программного обеспечения заключается не просто в более быстрой генерации кода. Это может быть помощь инженерным командам в понимании того, что фактически было создано, что остаётся незавершённым и где реализация отклонилась от задокументированного намерения.

Разница между сообщаемым прогрессом и готовностью продукта

Большинство инструментов планирования полагаются на статус, сообщаемый людьми. Разработчики обновляют рабочие элементы. Системные инженеры поддерживают требования. Инженеры по тестированию записывают результаты верификации. Лидеры проектов консолидируют эти входные данные в отчёты и панели мониторинга. Каждое действие необходимо, но результирующая картина только настолько точна, насколько синхронизированы лежащие в её основе артефакты.

В крупной программе SaMD соответствующая информация может быть распределена across:
- Системы управления требованиями и жизненным циклом приложений
- Репозитории исходного кода
- Документы по архитектуре и дизайну
- Файлы управления рисками
- Автоматизированные тестовые фреймворки
- Системы отслеживания дефектов
- Конвейеры непрерывной интеграции
- Инструменты гибкого планирования

Ни один артефакт не предоставляет полной картины готовности продукта. Исходный код может показывать, что возможность существует, но не то, была ли она реализована в соответствии с её требованиями. Система требований может показывать, что требование утверждено, но не то, завершён ли каждый поддерживающий путь кода. Система управления тестированием может показывать, что тестовые случаи пройдены, но не то,

invalidated ли недавние изменения кода лежащие в их основе предположения. По мере усложнения продукта ручное согласование этих представлений становится всё более трудным.

Переход от генерации кода к осведомлённости о реализации

Текущие обсуждения генеративного ИИ в разработке программного обеспечения часто сосредоточены на продуктивности: генерация кода, объяснение функций, создание модульных тестов или ускорение отладки.

Эти приложения полезны, но они представляют только один слой потенциальной ценности. LLM, которые могут анализировать репозитории, интерфейсы, зависимости, истории коммитов, результаты тестов и связанные рабочие элементы, также могут помочь командам ответить на вопросы более высокого уровня:
- Какие части клинического рабочего процесса реализованы?
- Какие предполагаемые поведения поддерживаются только частично?
- Существуют ли функции-заполнители, mocked интеграции или неполные пути обработки ошибок?
- Было ли управление рисками реализовано across каждого соответствующего пути выполнения?
- Соответствуют ли доказательства верификации последнему состоянию кода?
- Какие требования, по-видимому, имеют изменения кода без обновлённых тестов?
- Какие закрытые рабочие элементы всё ещё имеют неразрешённые зависимости реализации?

Эту возможность можно описать как **осведомлённость о реализации**: способность оценивать фактическое состояние программного обеспечения относительно предполагаемого поведения системы.

Это не заменяет формальную прослеживаемость, верификацию или инженерное суждение. Вместо этого это предоставляет ещё один источник доказательств, который может помочь командам раньше выявлять несоответствия.

Пример SaMD: когда «Готово» не означает «Завершено»

Рассмотрим гипотетическое приложение для удалённого мониторинга сердца, предназначенное для анализа данных электрокардиограммы и уведомления клиницистов при обнаружении потенциально значимой аритмии. На уровне планирования возможность может выглядеть как единая функция: Обнаружить подозреваемую аритмию и представить событие клиницисту для рассмотрения.

В реализации, однако, эта возможность может зависеть от нескольких элементов:
- Получение данных ЭКГ от подключённого устройства мониторинга
- Подтверждение полноты данных и качества сигнала
- Предварительная обработка сигнала перед анализом
- Запуск алгоритма обнаружения аритмии
- Присвоение классификации события или оценки достоверности
- Применение правил приоритизации оповещений
- Представление события в панели клинициста
- Предоставление клиницисту возможности рассмотреть, аннотировать или отклонить событие
- Обработка задержанных, дублированных или повреждённых данных
- Запись истории обнаружения и рассмотрения
- Верификация того, что связанные с безопасностью элементы управления работают как задумано

Предположим, что конвейер обработки сигналов, алгоритм и основная панель реализованы. Большинство задач разработки закрыты, и функция выглядит почти завершённой в плане проекта.

Инженерный агент с поддержкой LLM, исследующий код, интерфейсы, доказательства тестов и связанные требования, может выявить иную картину.

Например, он может определить, что:
- Проверки качества сигнала выполняются во время приёма данных, но обходятся во время ручной повторной обработки файлов
- Интерфейс клинициста отображает оценку достоверности, но один путь кода всё ещё использует значение по умолчанию
- Сбой в службе уведомлений регистрируется, но не доводится до пользователя
- Автоматизированные тесты покрывают номинальное обнаружение аритмии,

но не условия ухудшенного сигнала
- Требование управления рисками, касающееся задержанного уведомления клинициста, связано с функцией, но не с завершённым тестом верификации

Возможность не просто «завершена на 80%». Что более важно, она содержит специфические пробелы, влияющие на безопасность, прослеживаемость и клиническую применимость. Это тот тип различия, который может обеспечить осведомлённость о реализации.

Почему вертикальное нарезание делает оценку на основе ИИ более полезной

Анализ на основе LLM становится более значимым, когда работа структурирована вокруг согласованных возможностей системы, а не изолированных технических компонентов.

Традиционная декомпозиция часто разделяет работу на категории, такие как фронтенд, бэкенд, база данных, алгоритм и тестирование. Это может упростить назначение ресурсов, но может затмить, существует ли usable, безопасная и тестируемая возможность across всей системы.

Вертикальное нарезание предлагает иную структуру. Вертикальный срез представляет собой сквозную единицу функциональности, которая охватывает слои, необходимые для доставки наблюдаемого поведения продукта. В SaMD это может включать:
- Пользовательский или клинический рабочий процесс
- Логику приложения
- Алгоритмы
- Управление данными
- Внешние интерфейсы
- Обработку ошибок
- Элементы управления рисками
- Деятельность по верификации

Когда артефакты планирования организованы вокруг вертикальных срезов, система ИИ имеет более чёткую единицу намерения, против которой можно оценивать реализацию.

Вместо определения того, содержат ли отдельные компоненты код, система может оценивать, существует ли полная возможность across необходимых слоёв.

В примере с мониторингом сердца соответствующий срез — это не просто «алгоритм аритмии» или «панель клинициста». Это полный путь от получения данных ЭКГ через обнаружение, приоритизацию, представление, рассмотрение клиницистом и запись события.

Эта структура также создаёт более сильную связь между планированием продукта и системной инженерией. Вертикальный срез может связывать клиническое намерение, реализацию программного обеспечения, управление рисками и доказательства верификации в рамках общей функциональной границы.

Как мог бы выглядеть рабочий процесс с поддержкой ИИ

Практическая реализация могла бы начаться с ограниченного консультативного рабочего процесса, а не позволять агенту ИИ автоматически изменять контролируемые записи.

Во-первых, организация определяет функциональный срез и связанные с ним доказательства. Срез может включать требования, критерии приёмки, элементы архитектуры, управление рисками, репозитории, интерфейсы и тесты верификации.

Система ИИ затем исследует доступные артефакты реализации и строит карту между намерением и доказательствами. Она может определить соответствующие исходные файлы, pull requests, тесты, определения интерфейсов и ссылки прослеживаемости.

Вместо присвоения единого процента завершения система производит оценку на основе доказательств. Например:
- Основная логика обнаружения, по-видимому, реализована
- Обработка качества данных несогласована across путей приёма
- Один сценарий сбоя оповещения lacks соответствующей обратной связи пользователю
- Связанное управление рисками не имеет завершённого результата верификации
- Автоматизированные тесты существуют, но не покрывают последнее изменение алгоритма
- Сквозной клинический рабочий процесс ещё не может быть продемонстрирован в условиях ухудшенной сети

Система могла бы затем опубликовать эти выводы в среде планирования, сгенерировать сводку для рассмотрения или пометить срез для человеческой оценки.

Улучшение обзоров дизайна и готовности к интеграции

Осведомлённость о реализации могла бы усилить несколько существующих активностей SaMD.

Обзоры дизайна

Перед формальным обзором анализ с поддержкой ИИ мог бы выявить расхождения между утверждёнными требованиями, артефактами реализации, управлением рисками и статусом верификации. Это позволило бы рецензентам сосредоточиться на неразрешённых технических вопросах, а не тратить время обзора на поиск отсутствующей информации.

Планирование интеграции

Команды могли бы определить функции, которые выглядят локально завершёнными, но остаются отключёнными от вышестоящих или нижестоящих компонентов. Это особенно ценно, когда приём данных, алгоритмы, облачные сервисы, пользовательские интерфейсы и внешние уведомления разрабатываются разными командами.

Реализация управления рисками

Система ИИ могла бы помочь locate пути кода, связанные с управлением рисками, определить, применяется ли управление последовательно, и выделить пути выполнения, где ожидаемое поведение отсутствует. Любой вывод всё равно потребует рассмотрения квалифицированными инженерами.

Готовность к верификации

Перед началом формального тестирования система могла бы сравнить текущую реализацию с покрытием тестов, требованиями и известными изменениями кода. Это может выявить отсутствующие условия тестирования, устаревшие предположения тестов или требования без соответствующих доказательств.

Анализ влияния изменений

Когда разработчик изменяет интерфейс, общий сервис, модуль обработки сигналов или клинический алгоритм, система могла бы определить затронутые возможности, тесты, требования и управления рисками. Это не заменило бы формальную оценку изменений, но могло бы сделать первоначальный анализ более комплексным.

Инструменты планирования должны отражать доказательства, а не заменять их

Агент ИИ мог бы в конечном итоге помочь синхронизировать планы проектов с реализацией. Однако позволять ему автоматически перемещать рабочие элементы в «завершено» было бы упрощением.

В SaMD завершение многомерно.

Код может быть завершён, в то время как верификация остаётся открытой. Верификация может быть завершена, в то время как обновления документации ожидают. Управление рисками может быть реализовано, но ещё не рассмотрено. Отдельный компонент может функционировать корректно, в то время как сквозной рабочий процесс остаётся недоступным.

Более полезная модель позволила бы системе отображать отдельные индикаторы для:
- Полноты реализации
- Полноты интеграции
- Прослеживаемости требований
- Покрытия управления рисками
- Готовности к верификации
- Статуса документации

Это предоставляет инженерным и программным лидерам более честную картину, чем единое поле статуса или процент завершения.

От статической прослеживаемости к живому представлению системы

Традиционная прослеживаемость устанавливает отношения между требованиями, дизайном, реализацией, рисками и тестами. Эти отношения остаются фундаментальными для регулируемой разработки.

Однако прослеживаемость часто оценивается как статическая структура: ссылки существуют, записи присутствуют, и требуемые поля заполнены.

ИИ с осведомлённостью о реализации создаёт возможность для более динамичного представления.

Система могла бы непрерывно спрашивать, остаются ли связанные артефакты согласованными друг с другом:
- Реализует ли текущий код всё ещё требование?
- Тестирует ли тест всё ещё соответствующее поведение?
- Ввело ли архитектурное изменение новую зависимость?
- Присутствует ли задокументированное управление рисками в каждом соответствующем пути кода?

Это перемещает организации от поддержания сети прослеживаемости к поддержанию живого представления состояния продукта.

Взгляд вперёд

Ценность ИИ в разработке SaMD не должна измеряться только тем, насколько быстро он генерирует код. Для сложного и критичного для безопасности программного обеспечения более значимым применением может быть помощь командам в понимании взаимосвязи между клиническим намерением, реализацией, риском и верификацией.

Долгосрочная возможность — это не система планирования, которая обновляется сама без участия человека. Это инженерная среда, которая непрерывно сравнивает то, что организация намеревалась создать, с тем, что, как показывают доказательства, фактически было создано — и помогает командам разрешать разницу до того, как она станет риском продукта на поздней стадии.

Перевод выполнен: 04.08.2026 | ai4med.ru

Машинный перевод. Рекомендуем сверять с оригиналом при клиническом использовании.