Тестировщик и DevOps: позволяют ли интерфейсы системы эффективно решать инциденты?

Максим Цепков

IT-архитектор и бизнес-аналитик

Навигатор и эксперт по миру Agile,

бирюзовых организаций и Спиральной динамике

http://mtsepkov.org

SQA Days-22 в Санкт-Петербурге

17-18 ноября 2017

Нужно ли решать инциденты – позиция компании

☹ Мы сделали систему, вы ее купили «как есть».
Мы исправляем ошибки, а все остальное – Ваши проблемы.

☺ Мы не только поставляем систему, но и внедряем ее и нам важно эффективное решение инцидентов службой сопровождения

☺ Мы не просто поставляем систему – мы поддерживаем бизнес клиентов, помогая решить его проблемы в нестандартных ситуациях

Принцип: **бизнес должен работать**, несмотря на то, что функционал системы ограничен и в ней есть ошибки – надо прорабатывать сценарии для нестандартных ситуаций

О каких инцидентах будет рассказ?

Отражение сделок в системе

- Произошла не стандартная сделка, которую надо отразить в системе
- При обработке сделки нашли ошибку, надо ее обойти
- Отчет надо поправить и отослать немедленно, а потом уже штатно

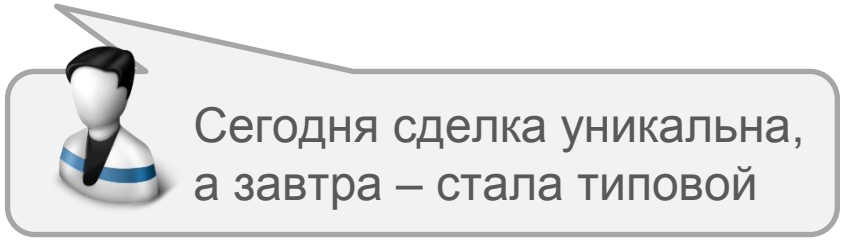
Интеграция с другими системами

- При обработке сообщения обнаружили ошибку в системе
- Повторно отправить сообщение другой системе для обработки
- Отправить в систему объект целиком, а не изменения к нему
- Большой объем изменений надо передать вне обычной интеграции

Инциденты работы со сделками

Зачем отражать нестандартные сделки в системе?

- Отражение в отчетах
- Внешний документооборот
- Отклонения касаются малой части сделки и есть желание воспользоваться функционалом системы для всего остального



Можно ли отразить любую сделку?

- Нет, есть принципиальное ограничение – структура базы данных
- Для большинства ситуаций структура БД – **не проблема**
  - Надо преодолеть контроль бизнес-логики при заведении и workflow сделки
  - Надо иначе создавать исполнительные документы или отражать в учете

Разработчик может просто поправить базу вручную – но это затратно и несет потенциальные проблемы

Что нужно от интерфейса системы?

- Функциональные возможности
  - Точечно отключаем ограничения и контроли бизнес-логики
  - Вручную создаем подчиненные документы – сообщения, платежи, проводки…
- Эргономика решения может быть ограничено
  - Рассчитано на опытных пользователей или инженеров поддержки третьей линии
  - Могут быть композитные решения: разработчик создал скрипт и его подключили
- Для проверки – возьмите сложные сделки, которые «пока не делаем»
  - Проработайте этот сценарий еще на этапе постановки
  - И проверьте в ходе тестирования
  - Не все сделки надо проверять, но нужен план, что делать, если они появятся
- И покажите эти сценарии архитекторам, чтобы они учли их при проектировании системы

Интерфейсы отчетов для нестандартных случаев

- Простой вариант – любой отчет экспортировать в Excel и поправить
- Более сложные – требуют отдельного проектирования
  - Фиксируем правки к отчету отдельно от вычислимых полей
  - Реализуем сравнение отчетов
- Композитный вариант: разработчик пишет запрос, который можно быстро подключить к системе в виде, доступном для пользователей

По опыту – очень эффективный способ для любых нестандартных отчетов

Инциденты интеграции

Работа с ними особенно важна, если смежная система – закрытая и недоступная для доработок, как свойственно legacy. Тогда надо обеспечивать сервис на нашей стороне

Интерфейсы приема – для решения инцидентов

Принцип: Ошибка приема сообщения – не всегда ошибка для бизнеса

Стратегия: эффективная диагностика и обработка внутри системы

- Сообщение сохраняется, даже если при обработке возникли ошибки
- Можно посмотреть структуру объектов сообщения, а не только xml
  - Оцените, сколько времени займет инженера при ручном разборе сообщений
- Можно запустить сообщение на повторную обработку
  - Протокол должен быть рассчитан на отсроченное уведомление об успехе
- Можно вручную провести требуемые действия с объектом
  - И пометить ошибку как «исправленную вручную»

Интерфейсы отправки для решения инцидентов

Сценарий: причину инцидента диагностировали и устранили, теперь надо обеспечить прохождение документа по интеграции

- Повторно отправить сообщение другой системе для обработки
- Отправить в систему объект целиком, а не изменения к нему
  - В том числе другой объект – возможно, синхронизация нарушилась по нему

Общий функционал интерфейсов

- Уведомления об ошибках или о том, что слишком долго нет ответа – чтобы оперативно узнавать о проблемах у бизнеса
- Просмотр журналов обмена и инцидентов, включая анализ и поиск не только во сообщениям, но и по передаваемым объектам – чтобы вести анализ инцидентов, выявляя системные проблемы

Альтернативные интерфейсы интеграции

- Ситуации массовой передачи данных
  - Начальная инициализация интеграции по объектам
  - Массовые изменения объектов при изменении структур данных или логики
  - Массовый перевод объектов в другую систему-владельца
  - И многое другое – из-за ограниченного функционала или пропускной способности интеграционной шины
- Массовая обработка должна выполняться **технологично**, а ошибки должны быть доступны для разбора через интерфейс интеграции

Заключение

- Быстрое получение MVP часто предполагает реализацию **только** основных успешных сценариев работы системы
- Однако, для бизнес-процесса необходима обработка технологичная особых ситуаций и сделок и эффективная работа инцидентов
- Принципиальная возможность должна быть заложена **в архитектуру**
- Сценарии использования и тест-кейсы необходимо прорабатывать – и показывать архитекторам, это хорошее начало обсуждения
- Все, что не заложено в систему придется делать разработчикам вручную – надо иметь свободных разработчиков на поддержке

Вопросы? Обращайтесь!

Максим Цепков http://mtsepkov.org