Борис Фищук («АРТ-Финтех»): Архитектурный подход и управленческий диктат

В гостях у «Б.О» постоянный автор нашего технологического раздела — Борис Фищук, один из основателей и лидеров группы компаний «АРТ-Финтех»



— Борис, в прошлых интервью мы много говорили об архитекторах АБС «АРНУВО». Вы сказали, что ни одно важное решение в компании не обходится без их участия. Упомянули в связи с этим словосочетание «управленческий диктат». Хотелось бы узнать об этом подробнее. Но сначала вопрос: что новенького?

— В первую очередь, упомяну наше пилотное внедрение АБС «АРНУВО».

— В каком банке оно идет?

— Мы предпочитаем называть только тех клиентов, с кем завершены внедрения.

— Ок! Признайтесь, не все просто с первым внедрением?

— А как же! Первое внедрение АБС — может ли оно быть простым?

— То есть микросервисы в бою оказались не так уж хороши?

— О, нет! Работают как надо! Мы ведь уже запустили несколько непростых и достаточно нагруженных проектов в нескольких банках и платежных системах на новой платформе!

— Так в чем же трудности?

— Сложность в том, что АБС была перепроектирована с нуля, и ничего, кроме идей, из наших старых систем взято не было! В том числе вся отчетность регулятора, множество интеграций и прочее.

— Зачем же так радикально?

— Увы, мы не видим способа перенести код из старых систем в новую. Натянуть Oracle на Postgre — это вивисекция похуже, чем с той самой совой и глобусом.

— Не сработает?

— Может быть, для очень мелких банков… посмотрим, как пойдут дела у конкурентов.

— Вы целитесь только в крупные банки?

— Крупные и средние — с приличной транзакционной нагрузкой.

— Хорошо. Перейдем, собственно, к архитектуре. Микросервисы — они и в Африке…

— Не уверен, что они водятся в Африке… У нас они вполне цивилизованные, можно сказать — ходят строем. «Африканские микросервисы» бегают по саваннам хаотично. Каждый из них может взаимодействовать с каждым…

— Ну, это какой-то пещерный век. В наше время микросервисы общаются через брокеры сообщений типа Kafka. Разве нет?

— Конечно, да, но что это принципиально меняет? Все равно каждый микросервис знает своих соседей и заботится о том, чтобы вызвать нужного и правильно ему скомандовать то, что считает нужным.

«Африканские микросервисы» — это парадигма хореографии. Каждый папуас исполняет свой танец и пытается заставить соседей танцевать его же. Но у них — свои танцы. Африканцы наделены потрясающим чувством ритма — и они танцуют один танец. А ритм задает кто?

— Вероятно, шаман со своим барабаном или бубном?

— Именно! Эта парадигма называется оркестровкой.

— Походит на оркестр с обязательным дирижером…

— Именно! Представьте, что музыканты начнут договариваться друг с другом: когда играть те или иные ноты. А если в оркестре 100 человек, да еще с совершенно разными инструментами?

— И кто же выступает дирижером в вашем «оркестре микросервисов»?
— Тоже микросервисы, только интеграционные.
— И в чем же упрощение?
— В подходе к их созданию. В нашей АБС запрещено писать интеграционные микросервисы в коде. Они всегда рисуются в BPM-нотации.
— Но ведь BPM — это медленно и ломко!
— А вот теперь мы близки к разгадке! BPM-движок — это таки-да медленный и ненадежный интерпретатор! Но мы используем лишь BPM-нотацию!
— Нарисовали модель процесса — и что потом?
— А потом нажали кнопку «Компилировать» — и BPM-модель превращается в микросервисы в исходном коде!
— И не надо править код руками?
— Можно, но у нас категорически запрещено! И затем — как можно давать аналитикам править исходный код? Эта дорога в ад.
— Вы хотите сказать, что большая часть кода системы написана автоматически?
— Так оно и есть! Моделируют процессы на BPM именно аналитики! А код создается автоматически!
— Уволим всех разработчиков?
— Вовсе нет! Кто же будет писать код внутри элементов BPM? Просто разработчики должны писать важные элементы системы. А для создания 95% процессов АБС аналитики подходят как нельзя лучше — процессы повторяются с незначительными изменениями!
— Речь идет о процессах наподобие кредитного конвейера?
— Далеко не только! Мы применили BPM-моделирование везде, где есть последовательности действий. Интеграции, схемы порождаемых транзакций, бухгалтерских проводок и даже авторизационные схемы для Нео-процессинга!
— О Нео-процессинге вы расскажете нам в следующий раз, а пока вернемся к бизнесу. Вы хотели рассказать о том, как связан архитектурный подход и управленческий диктат.
— В прошлой статье («Б.О» № 3) я высмеивал популярную в банках идею о том, что «надо дать разработчикам свободу — и они сделают как лучше».
— Лучше — кому?
— Вы смотрите в корень проблемы! Лучше для разработчиков — не означает лучше для банка.
— Так ли ужасен хардкод, который вы критикуете?
— Вовсе нет! Любой большой компании и банку нужны серьезные разработчики! Речь лишь о том, чтобы дать им возможность заниматься действительно серьезными делами, а автоматизацию рутинных бизнес-процедур перекинуть на аналитиков!
— Мы знаем множество неудачных примеров использования фреймворков в серьезных энтерпрайз-решениях. Результаты получались совсем не те, которые ожидали энтузиасты этих систем. Почему?
— А вот тут мы переходим ко второй части нашей сегодняшней беседы — управленческому диктату в разработке. Вместо правильного развития фреймворка очень часто останавливались на середине: вот, он помогает сформировать прототип решения, а остальное хардкодом поправим!
— И через полгода таких правок фреймворк превращается в тыкву! Ваше решение?
— Мы строго разделяем области ответственности разработчиков и аналитиков. Разработчики лишь пишут код внутри элементов BPM, но не моделируют в нем. Аналитики строят BPM-модели, но ни в коем случае не пишут код! Потом аналитик нажимает волшебную кнопку «Компилировать», и в итоге мы получаем готовый микросервис на TypeScript или Java.
— А затем берем в руки напильник и...
— А вот и не угадали! Править итоговый код микросервиса в нашей компании строго запрещено! Если есть ошибки в коде блоков BPM, они правятся прямо там — и процесс повторяется.
— Почему так?
— Потому что, по нашему глубокому убеждению, знания, зашитые в код, уже не являются в полной мере собственностью компании или банка.
— Они становятся немного личной собственностью программиста?
— Немного — да. А если таких знаний — весь код системы? Мы попадаем в зависимость от разработчиков, сами того не замечая: от качества их работы, их знаний, личных предпочтений и даже настроения.

А еще мы хотим обратить внимание коллег на то, что BPM-нотация является самодокументируемой! Любой аналитик читает ее как открытую книгу и может быстро найти ошибки или внести изменения.

И наконец очень важный аргумент. Как известно, на каждом звене передачи информации между людьми теряется до 30% информации. Разрабатывая системы в BPM-подходе, мы исключаем лишние звенья из цепи — получив задачу, аналитик буквально сразу начинает рисовать ее решение!
— И запретив изменять код вручную, вы экономите на многих видах тестирования?
— Точно! Нам не нужно ловить простые человеческие описки/опечатки в коде и проводить код-ревью — это всего лишь машина, которая делает ровно то, что ей сказано. Код всегда одного качества!
— Ок, а вы не используете AI в своей работе, например, при генерации кода?
— В настоящий момент необходимости нет, но инструментарий активно изучаем. Как и все, мы впечатлены бурным развитием AI и без внимания его не оставляем!
— Вы хотите оставить существующую 100% определенную генерацию кода (все же это деньги!), но думаете над возможностью строить процессы из ваших текущих элементов BPM при помощи AI-промптов?
— Вы правильно поняли ход нашей инженерной мысли, но... пока это секрет! Б.О

реклама