26 сентября 2024

Как не превратить корпоративную ИТ-систему в «монстра»

Почему монолитные корпоративные системы — это бомба замедленного действия и как избежать ошибок при проектировании?

Крупные и средние компании часто сталкиваются с ситуацией, когда корпоративная информационная система (например, высоконагруженная учетная система) перестает устраивать и бизнес, и пользователей. Такие системы-«монстры» ежедневно требуют все больше усилий для поддержания их работоспособности, финансовых вливаний на доработки и ресурсов для очередных обновлений. Но часто все эти усилия не дают нужного эффекта, который так ждет бизнес. Расскажем, как избежать такой мутации системы и как не допустить ситуации, когда она превращается в «монстра».

ОТ МОНОЛИТА К ГИБКОСТИ

Еще недавно на рынке программного обеспечения преобладала монолитная модель ИТ-архитектуры для систем и бизнес-приложений. Модель монолитной архитектуры — это устоявшийся классический подход к созданию ПО, который предполагает наличие единого блока, функционирующего самостоятельно или изолированно от других приложений. В таком случае все функции и модули системы или приложения

работают как единое целое и обновить отдельный компонент независимо невозможно — для этого потребуется обновление или изменение всей системы. Однако сейчас на рынке набирает популярность гибкая и масштабируемая архитектура: по нашим наблюдениям, компании все чаще переходят к микросервисам, которые подразумевают метод организации архитектуры, основанный на ряде независимо развертываемых служб, и позволяют быстро вносить необходимые изменения. Согласно опросу Cnews, такую архитектуру уже используют 35% российских компаний, а еще 10% планируют внедрить ее в ближайшее время. Вместо единого блока применяются отдельные модули для загрузки информации, работы с устройствами сбора данных, генерации отчетов. Модули запускаются в нескольких экземплярах и располагаются на различных серверах — так можно повысить скорость работы и надежности системы. Но даже это не дает гарантии, что несколько плохо связанных между собой бизнес-приложений, построенных по принципу микросервисов, со временем не превратятся в «монстра».

КАК РАСПОЗНАТЬ «МОНСТРА»?

Затраты на поддержку системы увеличиваются, пользователи ежедневно забрасывают техподдержку десятками заявок о сбоях системы, а генеральный директор не сумел собрать отчет для инвесторов, потому что система снова «прилегла», не выдержав нагрузки. Знакомая ситуация?

Первый тревожный сигнал для бизнеса и пользователей — медленное внесение даже небольших изменений и доработок. Например, бизнес-процесс в компании изменился после открытия нового филиала, однако в корпоративной учетной системе никаких изменений не произошло: доработки занимают месяцы, а обоснование экономической эффективности таких доработок становится трудной задачей.

Обратите внимание на то, как работают в системе новые сотрудники. Если новым коллегам трудно понять принцип работы, в простых операциях часто допускаются ошибки, последовательность действий запоминается не с первой попытки, а на составление и настройку отчетов тратится много времени — это тоже тревожные симптомы. «Красными флагами» является медленная работа или зависание системы, а также дублирование операций, когда для выполнения одной и той же задачи пользователь вынужден вводить одни и те же данные вручную. Другой пример — когда система не может корректно обработать запрос пользователя с первого раза и требует повторного выполнения операции.

Оцените работу ИТ-команды. В случае если для разработчиков настойчивые пожелания бизнеса устранить критичные недочеты в работе системы означают изменения на уровне архитектуры, а настройка новых процессов требует больших вложений или растягивается на неопределенные сроки, то это повод задуматься. Еще одним сигналом может быть ситуация, когда новые сотрудники ИТ-подразделения долго вникают в нюансы архитектуры и кода, а когда обращаются к технической документации и базе знаний, то не находят ответа или подсказки, потому что базу не вели и не обновляли последние

пару лет.

ПОЧЕМУ СИСТЕМЫ ПРЕВРАЩАЮТСЯ В «МОНСТРА»?

По нашему опыту, наиболее распространенная причина превращения системы в «монстра» — ошибка проектирования на старте внедрения.

Представьте ситуацию: менеджмент компании N принимает решение о приобретении и настройке системы управления взаимоотношениями с клиентами (CRM). Команда определила конкретные цели, результаты, функции, задачи, стоимость и сроки проекта. После того как поставщик провел первые демонстрации прототипов CRM-системы, стейкхолдеры начинают высказывать дополнительные пожелания. Коммерческий директор предлагает добавить в систему блок складского учета для скоропортящихся товаров. Маркетолог просит реализовать интеграцию с сервисами электронных рассылок и поддержку программ лояльности для постоянных клиентов. Технический директор предлагает разработать гибкий конструктор коммерческих предложений и настроить маршруты согласований в системе электронного документооборота. Директор по развитию бизнеса планирует открыть филиал в Китае через полгода и просит сделать CRM сразу с поддержкой английского, а еще лучше — китайского языка.

Часто на практике на этапе разработки и проектирования системы в нее включаются бизнес-требования, которые не подкреплены техническими

возможностями, проще говоря, это пожелания бизнеса, идущие вразрез с первоначальным назначением системы. В итоге проектируется перегруженная ненужными функциями система, далекая от первоначального назначения. Одни функции дублируются в смежных системах, другие не используются вовсе.

В подобных ситуациях необходимо фиксировать и классифицировать в едином бэклоге пожеланий идеи, с которыми приходит к разработчикам бизнес. Примеры критериев: по степени важности (критичные, важные, желательные), по типу изменений (функциональные, нефункциональные, технические).

Для каждого пожелания из «банка идей» не лишним будет рассчитать экономические эффекты и обосновать целесообразность доработок в той или иной системе. Часто уже на этом этапе становится понятно, какие функции действительно необходимы, а какие нет. Не забывайте, что бэклог требует регулярного пересмотра и обновления; периодичность обновлений зависит от скорости изменений требований к системе.

КАК НЕ СОЗДАТЬ «МОНСТРА»?

На этапе проектирования системы советуем привлекать enterprise-архитектора, а если такой роли в команде еще нет, то, как минимум

потребуются заключение эксперта из ИТ-функции компании, который понимает и видит полный ИТ-ландшафт предприятия. Задача enterprise-архитектора — помочь дуэту «бизнес-заказчик и разработчик» принять взвешенное решение, в частности, ответить на вопрос, в какой целевой системе имеет смысл реализовывать проект?

Если ИТ-система будет реализована в неправильной целевой системе, это непременно приведет к печальным последствиям. Во-первых, такая система может не соответствовать потребностям пользователей и не решать поставленные задачи. Во-вторых, такую систему непросто адаптировать для последующего масштабирования. Неправильный выбор целевой системы ведет к неизбежным сложностям в эксплуатации. Замена такой системы для бизнеса всегда непростая и дорогая задача.

Исходя из нашего опыта для проектирования системы enterprise-клиенты формируют архитектурные комитеты — коллегиальные органы, состоящие из специалистов в отдельных областях. Это системные архитекторы, разработчики программного обеспечения, тестировщики, эксперты в области защиты информации. Цель архитектурного комитета — спроектировать оптимальную архитектуру системы, учитывая требования заказчика, выбрать правильный стек технологий и целевую систему.

При принятии решения о выборе целевой системы советуем руководствоваться двумя принципами:

- Принцип единой ответственности. Каждый компонент или модуль системы выполняет определенный набор задач так, чтобы изменения одной части системы не приводили к необходимости изменений других частей.
- Принцип ясных границ между компонентами. Требуется четко понимать, где в проектируемой архитектуре заканчиваются границы одного компонента или модуля ИТ-ландшафта и начинаются границы и зоны другого модуля.

Такой подход помогает уменьшить сложность ИТ-системы, упростить тестирование и поддержку, повысить качество кода.

Классический пример из учебника. У системы бухгалтерского учета есть назначение — поддержка требований регулятора с минимальными затратами на обновления. В этом случае новые функции, которые бизнес хочет реализации в системе, проходят через фильтр назначения. Задавайте себе вопрос, позволяют ли новые функции работать системе по назначению или нет?

СУЩНОСТИ В СИСТЕМЕ

Другая распространенная ошибка при проектировании ИТ-систем — некорректное определение сущностей. Сущности — это объекты, события, процессы или другие элементы, которые учитываются при создании ИТ-системы. Они описывают структуру данных и отношения друг с другом, что

дает возможность разработчикам создавать логическую модель для базы данных и определять способы взаимодействия с этими данными через пользовательский интерфейс.

Разберемся, как ошибка в определении сущности может привести к созданию неэффективной системы. Предположим, в системе создается такая сущность, как «Пользователь» — объект, который взаимодействует с системой. Сущность «Пользователь» обладает конкретными характеристиками: логин, пароль и права доступа. Другая сущность — «Сотрудник», его характеристики: должность, место в организационной структуре компании. Причем «Сотрудник» в широком смысле — это «Физическое лицо», обладающее именем, адресом прописки и полом.

Теперь давайте представим ситуацию, что при проектировании системы эти три сущности соединились в одну — «Пользователя», который наделен такими атрибутами, как должность, подразделение, фамилия, дата рождения. Например, ИТ-партнер внедрил систему, где доступ к данным разграничивается на уровне подразделения и должности. Спустя некоторое время наш «Пользователь» получил повышение, атрибут «Подразделение» изменился, а вместе с этим пропал и доступ к нужным для работы данным.

Точечное исправление такой ошибки позволит решить проблему «здесь и сейчас», но обернется дорогостоящими доработками по внесению изменений на архитектурном уровне и ростом критической массы подобных правок, что в конечном счете приведет к деградации системы. Таких триад, сведенных к одной, много. Например, «Клиент» — «Юридическое лицо» —

«Торговая точка» или «Товар» — «Упаковка» — «Груз».

CHECKPOINT ИЛИ ВМЕСТО ПОСЛЕСЛОВИЯ

Для того чтобы избежать трансформации корпоративной системы в монолитную и неэффективную конструкцию, постарайтесь следовать этим правилам:

- Управляйте бэклогом доработок и требований к системе.
- Избегайте хаотичных требований к доработкам.
- Планируйте доработки и тщательно анализируйте поступающие требования к системе.
- Ведите подробную документацию и описывайте новые процессы.
- Думайте о будущем системы на старте проекта по внедрению, сразу закладывайте необходимый уровень гибкости.
- Выбирайте целевую систему для реализации доработок.
- Учитывайте реальные потребности бизнеса.

Если вы унаследовали от прошлой команды систему-«монстра», то можно обратиться к профессионалам, которые могут подтвердить опыт рефакторинга кода деградировавших систем и помогут с выбором инструментов для «лечения» системы. Универсальных рецептов тут нет — это всегда требует погружения в особенности проекта и глубокого анализа предпосылок и причин, приведших к такому результату. Хорошие новости —

при интенсивной «терапии» все точно поправимо.