Вопрос 2. Осуществление интеграции программных модулей и компонент и верификации выпусков программного продукта.

Интеграция – это очень важная часть работы по автоматизации бизнес-процессов, так как требуется она постоянно. В разных ситуациях возникает потребность оперативно обмениваться данными между различными конфигурациями программных продуктов, между ПО и сайтами в интернете, между ПО и системами информации. Достаточно часто требуется интегрировать между собой различные веб сервисы, например, интернет-магазин и CRM-систему. В общем, объединить работу различных подразделений компании и автоматизировать рабочий процесс без использования интеграции в большинстве случаев невозможно.

Интегра́ция (от лат. integratio — «соединение») — процесс объединения частей в целое. В зависимости от контекста может подразумеваться:

- Веб-интеграция — объединение разнородных веб-приложений и систем в единую среду на базе веб.
- Интеграция данных — объединение данных, находящихся в различных источниках и предоставление данных пользователям в унифицированном виде

Интеграция программных систем и продуктов — это обмен данными между системами с возможной последующей их обработкой.

1. Цели и задачи интеграции

Интеграция приложений — это стратегический подход к объединению информационных систем, который обеспечивает возможность обмена информацией и поддержания распределенных бизнес-процессов. Интеграция информационных систем дает предприятию такие несомненные конкурентные преимущества, как:

– ведение бизнеса в режиме реального времени с использованием событийно-управляемых сценариев;

– владение достоверной, полной и своевременно полученной информацией.

Задача интеграции — обеспечить эффективный, надежный и безопасный обмен данными между различными программными продуктами, изначально не предназначенными для совместной работы. Как правило, требования бизнеса эволюционируют быстрее, чем способы их поддержки информационными технологиями.

Основными движущими силами интеграции являются:

- электронный бизнес;
- интеграция унаследованных информационных систем, поддерживающих ключевую функциональность, с Web-приложениями (Web-сервисами и порталами) с целью получения доступа к бизнес-функциям через Интернет;
- управление цепями поставок — интеграция разрозненных систем управления заказами, MRP-систем, систем календарного планирования, систем транспортного менеджмента с целью прямого обмена информацией между покупателями и поставщиками в режиме реального времени;
- управление взаимоотношениями с клиентами — получение единого консолидированного представления о клиенте путем объединения данных о нем, распределенных между несколькими изолированными приложениями (интеграция клиентских баз данных, call-центров, интернет-сервисов);
- внедрение ERP: интеграция модулей поддерживающих базовую функциональность, со специализированным программным обеспечением, используемым организацией;

- электронное правительство — интеграция унаследованных backend систем с front-end Web-приложениями, организация обмена данными между правительственными учреждениями;

- самообслуживание клиентов: возможность самостоятельно выполнять действия, традиционно являющиеся функцией обслуживающего персонала, требует интеграции пользовательских приложений с back-end-системами;

- Business Intellegence — сбор данных из различных приложений и источников в хранилище данных с целью их обработки и анализа;

- управление знаниями — обеспечение доступа в режиме реального времени к корпоративному контенту, распределенному многочисленными источниками, с целью управления знаниями в масштабах предприятия;

- облачные технологии — интеграция существующих бизнес-приложений с облачными приложениями и сервисами;

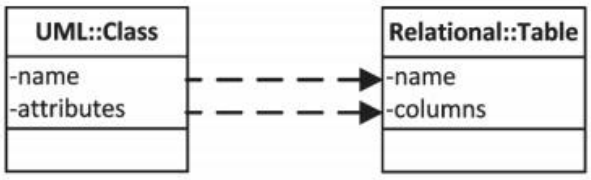
- аутсорсинг бизнес-процессов — интеграция с информационными системами партнеров.

Основные выгоды, которые предприятие может получить в случае успешной реализации интеграционного проекта:

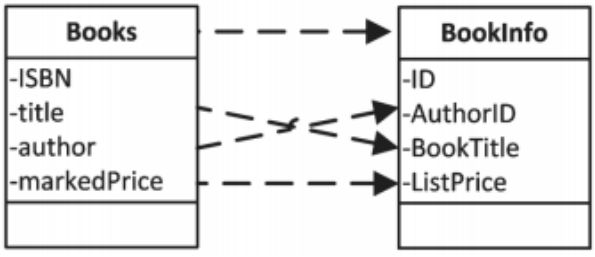
– улучшение качества поддержки и обслуживания клиентов;
– автоматизация бизнес-процессов;
– уменьшение производственного цикла;
– сокращение количества ошибок обработки данных;
– прозрачность процессов;
– уменьшение стоимости транзакций;
– оптимизация логистических процессов;
– более тесное взаимодействие с бизнес-партнерами;
– быстрое внедрение новых бизнес-сервисов;
– сохранение инвестиций в информационные технологии.

2. Уровни интеграции

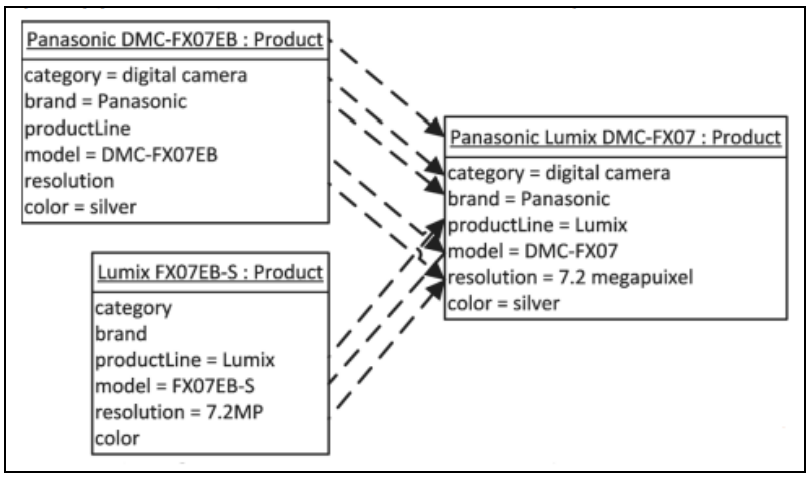
Интеграция моделей данных: сопоставление элементов моделей (языков)



Интеграция схем данных: сопоставление типов (структур) данных и их атрибутов



Собственно интеграция данных: преобразование исходных коллекций данных, элементов коллекций и значений их атрибутов



Завершение интеграции на более высоком уровне обычно является предусловием для интеграции на более низком уровне

3. Типы интеграционных решений

В зависимости от принадлежности объединяемых приложений выделяют:

1) интеграцию корпоративных приложений в пределах предприятия: автоматический событийно-управляемый обмен информацией между приложениями и системами, действующими на предприятии или в организации;

2) интеграцию приложений между предприятиями: автоматический событийно-управляемый обмен информацией между приложениями или системами нескольких взаимодействующих предприятий или организаций.

Как известно, информационные системы обеспечивают поддержку одного из трех уровней управления: операционного, тактического и стратегического.

В данном контексте существует два **варианта построения интеграционного решения:**

- **горизонтальная интеграция** — интеграция информационных систем или приложений, относящихся к одному уровню,
- **вертикальная интеграция** — интеграция приложений и систем, находящихся на различных уровнях информационной пирамиды.

Типичным примером горизонтальной интеграции является автоматизация управления цепями поставок (различные приложения или компоненты обеспечивают полный цикл логистических операций).

Наиболее часто встречающийся пример вертикальной интеграции — сбор данных операционных систем в единое корпоративное хранилище данных с целью их последующего использования для анализа, управления и получения консолидированной отчетности.

4. Виды интеграции программных модулей

Пошаговая интеграция заключается в том, что код разрабатывается и тестируется малыми компонентами, которые затем постепенно собираются в единое целое. Преимущества пошаговой интеграции:

- **Простота поиска ошибок.** При возникновении ошибки понятно, где ее искать: либо непосредственно в новом или измененном компоненте, либо в точке его

взаимодействия с системой. Кроме того, при пошаговой интеграции ошибки часто обнаруживаются последовательно, одна за другой, и это упрощает их поиск.

- **Компоненты тестируются в более полном объеме.** Интеграция компонентов осуществляется по мере их интеграции и тестирования. Благодаря этому работоспособность компонентов испытывается более часто, чем если бы интеграция проводилась в один заход.
- **Компоненты быстрее вводятся в строй.** Разработчики быстрее видят результаты своей работы, и им не приходится ждать завершения разработки всей системы. Это положительно влияет на мотивацию. Кроме того, это позволяет быстрее получить первые отзывы.

Противоположный подход к интеграции называется **поэтапной интеграцией**. Принцип поэтапной интеграции заключается в одновременной интеграции нескольких (новых и измененных) компонентов.

Основной недостаток поэтапной интеграции заключается в том, что в нее вовлечено много переменных, и это затрудняет поиск ошибок. Ведь ошибка может находиться не только в любом из новых компонентов, но и во взаимодействии новых компонентов с ядром системы и во взаимодействии между новыми компонентами.

Компонентно-ориентированный подход – создан для проектирования и реализации крупных и распределенных программных систем (корпоративных приложений). С точки зрения КОП, программная система – это набор компонентов с четко определенным интерфейсом. Изменения в систему вносятся путем создания новых компонентов или изменения старых. Наследование реализации запрещено. Наследуется только интерфейс.

Программный компонент – это автономный элемент программного обеспечения, предназначенный для многократного использования, который может распространяться для использования в других программах в виде скомпилированного кода. Подключение к этим программам осуществляется с помощью интерфейсов компонента. Взаимодействие с программной средой осуществляется посредством инициации и реагирования на события.

Преимущества использования компонент:

- Оптимизация стоимости и скорости разработки программной системы за счет использования готовых блоков.
- Повышение эффективности повторного использования кода.
- Повышение масштабируемости программной системы

К разработке программных компонентов предъявляются серьезные требования:

- полная документированность интерфейса;
- тщательное тестирование;
- тщательный анализ входных значений;
- возврат адекватных и понятных сообщений об ошибках;
- необходимость предусмотреть возможность неправильного использования.

Цель применения распределенных компонентных технологий:

- интеграция сервисов для приложений на базе различных платформ;
- обеспечение прозрачного механизма взаимодействия и обмена данными между элементами.

5. Этапы интеграции ПО

Смысл интеграции заключается в том, чтобы данные, которые пользователь вводит в одну систему, автоматически переносились в другую. Продукт, в который пользователь вводит данные, называется источник. А получатель данных, соответственно, приемник.

Достаточно часто данные переносят в обе стороны, например, после преобразования в системе-приемнике результаты отправляются обратно в источник. А потому интеграция бывает как односторонней, так и двухсторонней.

Например, *если вы объединяете конфигурацию 1С: Торговля с 1С: Бухгалтерией, вам может потребоваться передать данные по всем продажам в бухгалтерию, а обратно получить сведения об оплате по этим продажам.*

Такой процесс интеграции можно разделить на простые этапы:

1. Определяем, какой продукт является источником, какой – приемником.
2. Сопоставляем объекты между источником и приемником.
3. Выбираем протокол для интеграции
4. Проводим постобработку данных (после переноса в одну из сторон)

Этот подход помогает работать системно, не упустить ни одного важного момента и провести интеграцию таким образом, чтобы клиенту было удобно работать в объединенной системе.

Важно: при интеграции различных программных решений нужно хорошо понимать их функционал. Изучать программный продукт в процессе интеграции – это не совсем корректно. При таком подходе чаще всего возникает множество ошибок и проблем, например, перенос не тех данных или сбои в работе. Поэтому сначала нужно хорошо изучить программный продукт, понять, что он может, каким образом в нем реализованы те или иные функции, и только потом заниматься интеграцией.

В принципе, в процессе интеграции вам может потребоваться и более сложный обмен, и придется вводить, например, трех- или четырехстороннюю интеграцию. Принципиальных отличий у односторонней, двусторонней или многосторонней интеграции не существует. Суть процесса остается прежней, просто в разные моменты времени приемник и источник меняются ролями. Единственное важное правило: при двухстороннем обмене необходимо хранить и дублировать уникальный идентификатор для всех систем, которые участвуют в интеграции.

1. Выбираем источник и приемник

Для каждого случая интеграции данных важно четко определить, какая система будет источником, а какая – приемником. Нужно понимать, в какой системе пользователь будет вводить данные, а какая станет получателем этих данных через интеграцию. Это обязательно согласовывается с клиентом (пользователем), кроме случаев, когда источник очевиден. при этом обязательно нужно поставить в известность клиента, что данные определенного типа следует вводить именно через систему-источник.

2. Сопоставление объектов (данных)

Каждый раз при работе с данными нужно очень хорошо понимать, что именно вы выгружаете, в каком виде, а также, куда вы будете выгружать эти данные. В некоторых случаях в источнике у вас будет строковая переменная, а в приемнике – два или более объектов. В других важно просто правильно выбрать объект-приемник.

Программист должен четко понимать, какие данные будут брать из источника, куда их нужно переносить, и как они будут обрабатываться. Также очень важно понимать, какие преобразования потребуются для выгружаемых данных.

Например, *нужные для интеграции данные в источнике хранятся в качестве перечисления в виде текста. А в приемнике (пусть это будет 1С) аналогичное перечисление имеет ссылочный тип. Следовательно, вам потребуется преобразовать текст в ссылку, и уже ссылку передать в документ.*

Требуются правила сопоставления, которые вы должны четко продумать и прописать. Более того, об этих правилах необходимо оповестить ваших клиентов. Важно понимать, что клиент не видит логику работы обмена данными, он не понимает особенностей интеграции.

Конечно, вы обязательно введете ограничение прав доступа, добавите другие варианты защиты. Но, как показывает практика, это не гарантирует от того, что пользователь совершит ошибку, из-за которой интеграция перестанет работать или будет работать не корректно. Это может быть кто-то из сотрудников, обладающий правами администратора, или приглашенный специалист, который дорабатывает, например, печатную форму документа, но при этом не осведомлен об особенностях интеграции.

В результате возникают самые разные казусы. Например, вы используете в качестве ключевого слова для поиска при сопоставлении слово «дилер». Клиент по каким-то причинам меняет его в программе-источнике на слово «дилеры». Казалось бы, мелочь! Но эта мелочь приведет к тому, что поиск в 1С перестанет работать.

Решение проблемы:

1. Обязательно оставлять клиенту подробно описанные правила сопоставления и пояснения, какие параметры и данные менять недопустимо.
2. Предусматривать варианты оповещения об ошибке. Т.е. не только фиксацию проблемы в логе ошибок, но и оповещение пользователя о сбое каким-то образом: при помощи SMS, письмом на email, всплывающими уведомлениями в 1С. А иногда всеми этими способами сразу.

Также стоит лог-файл ошибок вести максимально подробно и как можно дольше хранить историю. Не забывайте, что вы имеете дело с данными, которые имеются в одной базе данных, но отсутствуют в другой. И без подробного отчета вам будет очень сложно понять, что именно произошло в процессе передачи данных.

3. Выбор протокола обмена данными: можно разрабатывать решение под заказчика, либо выбрать типовой обмен данными. Как правило, типовые продукты всегда сильно перегружены возможностями, которые вашему клиенту не нужны и в результате процесс обмена значительно замедляется, а число возможных ошибок вырастает в разы. Кроме того, при выборе типового программного решения вы очень сильно зависите от поставщика программного обеспечения. Для любого исправления бага вам придется ждать выпуск очередной версии программы. Также придется подстраиваться при обновлениях под все изменения в работе, который внес разработчик.

Потому при выборе между самостоятельным написанием обмена данными и типовым решением, которое не на 100% подходит для данной ситуации, лучше писать обмен самому.

Выбор конкретного протокола обмена данными **(**REST API, SOAP или прямое подключение к базе приемника) в большинстве случаев напрямую зависит от системы, которую вы интегрируете. В большинстве случаев программисту приходится учитывать требования обеих систем, поэтому, если система может работать с несколькими протоколами, выбирайте тот, который вам удобнее.

В некоторых случаях возникают проблемы клиентского доступа и обмен не работает. Самые распространенные ситуации:

- Ограничение доступа по IP.
- Ограничение прав пользователя.
- Ограничение по количеству обращений к источнику или приемнику

В первых двух случаях ограничения обычно связаны с политикой информационной безопасности предприятия, и решаются они на административном уровне. Для пользователей, которым потребуется работа с обменом данных, системный администратор настроит перечисленные вами права. Аналогично для ограничения по IP. В случае работы с CRM-системой ограничения обычно обусловлены оплаченным пакетом услуг. Здесь достаточно оповестить клиента о наличии такого ограничения, и, при необходимости, помочь оплатить и настроить расширенный пакет.

При интеграции с 1С очень часто ошибки обмена данных возникают из-за неверного выбора УИ (уникального идентификатора). Суть проблемы заключается в том, что объекты в 1С имеют два типа УИ: один уникален внутри выбранного типа объектов. Второй используется для работы со всей базой данных. Если вы будете проводить поиск по всему справочнику с использованием идентификатора, который предназначен для работы внутри определенного типа данных, возникнет ошибка. Объект может быть вообще не найдет, либо система найдет сразу несколько разных объектов. К этой особенности 1С нужно относиться очень внимательно.

Для обмена данными используются самые разные форматы. Это может быть JSON, XML, CSV, TXT, прямой доступ к базе и т.д. Каких-то ограничений здесь нет, нужно исходить из рациональных требований проекта.

4. Постобработка данных требуется для того, чтобы полученные данные прошли полный жизненный цикл и приняли участие в каких-то последующих бизнес-процессах. А потому после загрузки должны запускаться оповещения или какие-то определенные процессы, например, обработка заказа. Кроме действий, которые нужно выполнить в приемнике, также часто требуется после завершения успешной передачи данных выполнить определенные действия в источнике. Что именно потребуется, зависит от требований проекта.

Например, *для интернет магазинов при интеграции чаще всего требуются:*

- *Оповещение менеджера о поступлении заказа, например, при помощи sms*
- *Уведомление пользователей о поступлении новых заказов или другой актуальной информации по email*
- *Звуковой сигнал и/или всплывающее окно в 1С с напоминанием о том, что появились новые запросы или заявки*

6. Интеграция данных и ее верификация

Процесс интеграции данных на практике представляет собой набор операций трансформации данных, представленных в SQL или подобном ему языке. Важным здесь является порядок исполнения операций.

Группы операций интеграции данных:

- Трансформация данных - преобразование из исходной схемы (схемы коллекции - источника данных) в целевую (единую интегрированную схему).
- Разрешение сущностей - выделение и связывание информации об одной и той же сущности реального мира из разных коллекций данных.
- Слияние сущностей - комбинация различных представлений одной и той же сущности реального мира в единое представление.

Верификация процессов интеграции – формальная проверка модулей и подсистем на соответствие требованиям. Верификация возможна, когда определена формальная семантика процессов интеграции.

Интеграция – это часть (иногда частный случай) внедрения программного обеспечения. И здесь, как и для любой другой работы по внедрению ПО, потребуется тестирование программистом, потом – лично консультантом, а также различные варианты тестирования вместе с пользователями.

Интегральное тестирование проверяет каждую сборку интеграции и каждую итерацию.

Один из способов планирования и выполнения интегральных тестов:

1. Решите, как и где хранить, повторно использовать и кодировать интегральные тесты. Отразите это в расписании проекта.
2. Выполните столько модульных тестов (снова), сколько позволит время, согласуя с контекстом сборки. Расставьте приоритеты тем модулям, в которых возможны ошибки
3. Выполните регрессионные тесты. Убедитесь, что существующие возможности системы не были нарушены.
4. Убедитесь, что требования сборки определены должным образом.
5. Выполните варианты использования, которые должны быть реализованы в сборке. Протестируйте на соответствие SRS.
6. Выполните системные тесты для этой сборки.

Варианты использования являются идеальным источником тестовых вариантов для интегральных тестов. Рекомендуется согласовывать каждый вариант использования с одной сборкой. Идея в том, чтобы варианты использования строились на основе уже интегрированных частей, тем самым, формируя представительные тесты использования программы.

Нет в ИТ более сложной задачи, чем интеграция систем. Параметры, отвечающие за сложность интеграции и варианты минимизации негативного влияния этих параметров:

- **Концептуальная разница** — основывается та том, что разработчики разных систем изначально приняли разные решения, предположения и допущения, которые концептуально не стыкуются между собой. Решается введением еще одного слоя абстракции, который концептуально не противоречит обоим подходам.

  При этом есть два варианта реализации:

  - когда получившаяся система становится централизованной, а две и более интегрируемых системы превращаются в подсистемы;
  - когда мы используем архитектуру брокера (посредника, не являющегося центром), при этом системы остаются независимыми, а брокер обеспечивает прослойку между ними.

- **Технологическая разница** — когда мы имеем несовместимые форматы обмена данными, протоколы взаимодействия и интерфейсы. Решается написанием конвертов, прослоек, брокеров и других примочек, не вполне красивых, но достаточно надежных.

- **Несовместимость лицензий –** решается индивидуально, на организационном уровне.

Литература:

1. Проектирование компьютерных систем.
Лекция 9. Интеграция, верификация и валидация системы /

http://elib.kstu.kz/fulltext/books/Proektirovanie_KS/teory/%D0%93%D0%BB%D0%B0%D0%B2%D0%B0%209.htm#_Toc518160197
2. Шемсединов Т. Интеграция информационных систем / https://habr.com/ru/post/117468/
3. Кинзябулатов Р. Интеграция программного обеспечения. Описание процесса от бизнес консультанта / https://habr.com/ru/company/trinion/blog/245615/
4. Ступников С. Верификация потоков работ интеграции данных / https://www.osp.ru/netcat_files/userfiles/Hadoop_TBD_2_2016/Stupnikov_tbd_2.pdf
5. Концепция: интеграция программного обеспечения / http://dit.isuct.ru/Publish_RUP/core.base_rup/guidances/concepts/software_integration_2F85C9B0.html