Вспомним этапы проектирования базы данных.

1. Системный анализ и словесное описание информационных объектов предметной области.

2. Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некой инфологической, например, ER-модели.

3. Даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой даталогической модели.

4. Физическое проектирование БД, то есть выбор способа размещения БД на внешних носителях и создание файла БД в какой-либо СУБД.

Анализ предметной области состоит из трех фаз:

1) анализ требований и информационных потребностей;

2) выявление информационных объектов и связей между ними;

3) построение модели предметной области и проектирование схемы БД.

Для анализа требований первой фазы удобно использовать диаграммы прецедентов (UML) и потоков данных (DFD).

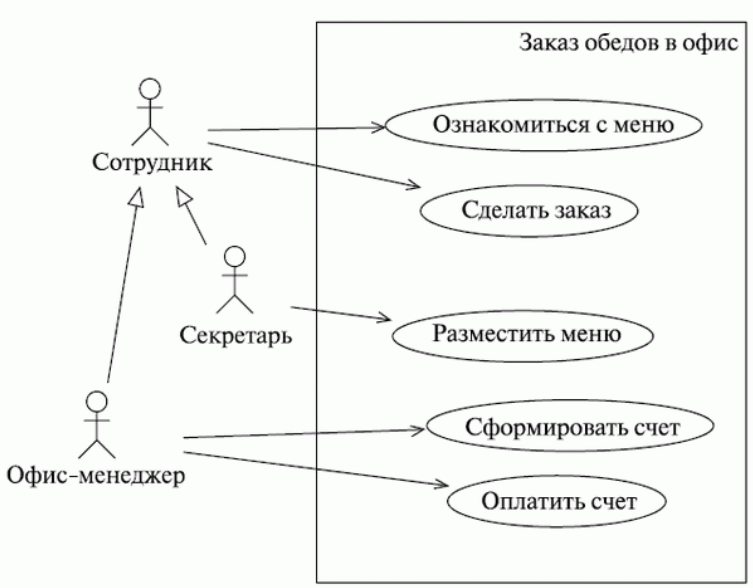
*Диаграммы прецедентов* составляют *модель прецедентов* (вариантов использования, use-cases). *Прецедент* - это функциональность системы, позволяющая пользователю получить некий значимый для него, ощутимый и измеримый результат. Каждый *прецедент* соответствует отдельному сервису, предоставляемому моделируемой системой в ответ на *запрос* пользователя, т. е. определяет способ использования этой системы. Именно *по* этой причине use-cases, или прецеденты, часто в русской терминологии фигурируют как *варианты использования*.

Рассмотрим пример *приложения* для автоматизации заказов обедов в *офис*. Секретарь размещает на сервере *меню* обеденных блюд на неделю. Сотрудники должны иметь возможность ознакомиться с *меню* и сделать заказ, выбрав блюда на каждый день следующей недели. *Офис*-менеджер должен иметь возможность сформировать счет и оплатить его.

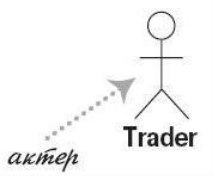
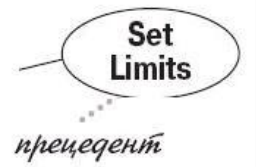
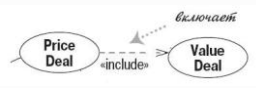
*Таблица* с описанием требований может быть, например, такой:

| **Прецедент** | **Действующее лицо** |
|---|---|
| разместить меню | секретарь |
| ознакомиться с меню | сотрудник, секретарь, офис-менеджер |
| сделать заказ | сотрудник, секретарь, офис-менеджер |
| сформировать счет | офис-менеджер |
| оплатить счет | офис-менеджер |

*Диаграмма прецедентов*, построенная на основе этой таблицы, может быть, например, такой:




Основной набор символов **диаграммы вариантов использования (прецедентов)** , необходимый для того, чтобы суметь прочитать диаграмму.

| **Термин** | **Изображение** | **Описание** |
|---|---|---|
| Сценарий | Вся диаграмма вариантов использования (ВИС) | Сценарий (scenario) – это последовательность шагов, описывающих взаимодействие пользователя и системы. |
| Актер |   | Актер (actor) представляет собой некую роль, которую пользователь играет по отношению к системе. |
| Прецедент |   | Обозначает выполняемые системой действия (могут включать возможные варианты), приводящие к наблюдаемым актёрами результатам. |
| include (включает) |   | Сложный шаг в прецеденте можно представить другим прецедентом. В терминах языка UML мы говорим, что первый прецедент включает (includes) второй. |
| Граница системы |   | Позволяет обозначить границы систем или подсистем. |

Прецеденты представляют собой ценный инструмент для понимания функциональных требований к системе. Первый вариант прецедентов должен составляться на ранней стадии выполнения проекта. Более подробные версии прецедентов должны появляться непосредственно перед реализацией данного прецедента.

Важно помнить, что прецеденты представляют взгляд на систему со стороны. А раз так, то не ждите какого-либо соответствия между прецедентами и классами внутри системы. Чем больше прецедентов на диаграмме, тем менее ценной кажется диаграмма прецедентов. Несмотря на то что в языке UML ничего не говорится о тексте прецедентов, именно текстовое содержание прецедентов является основной ценностью этой технологии.

Большая опасность прецедентов заключается в том, что разработчики делают их очень сложными и застревают на них. Обычно чем меньше вы делаете, тем меньший вред можете нанести. Если у вас немного информации, то получится короткий, легко читаемый документ, который явится отправной точкой для вопросов. Если информации слишком много, то вряд ли кто-то вообще будет ее изучать и пытаться понять.

Пример использования

**Прецеденты** – это технология определения функциональных требований к системе. Работа прецедентов заключается в описании типичных взаимодействий между пользователями системы и самой системой и предоставлении описания процесса ее функционирования. Вместо того чтобы описывать прецеденты в лоб, я предпочитаю подкрасться к ним сзади и начать с описания сценариев. **Сценарий** (scenario) – это последовательность шагов, описывающих взаимодействие пользователя и системы. Поэтому при наличии онлайнового магазина, основанного на веб-сайте, мы можем использовать сценарий ***«Покупка товара» (Buy a Product)***, в котором происходит следующее.

***Покупатель просматривает каталог и помещает выбранные товары в корзину. При желании оплатить покупку он вводит информацию о кредитной карте и производит платеж. Система проверяет авторизацию кредитной карты и подтверждает оплату товара тотчас же и по электронной почте.***

Подобный сценарий описывает только одну ситуацию, которая может иметь место. Однако если авторизация кредитной карты окажется неудачной, то подобная ситуация может послужить предметом уже другого сценария. В другом случае у вас может быть постоянный клиент, для которого проверка информации о покупке и кредитной карте не обязательна, и это будет третий сценарий.

Так или иначе, но все эти сценарии похожи. Суть в том, что во всех трех сценариях у пользователя одна и та же цель: купить товар. Пользователь не всегда может это сделать, но цель остается. Именно цель

пользователя является ключом к прецедентам: прецедент представляет собой множество сценариев, объединенных некоторой общей целью пользователя.

В терминах прецедента пользователи называются актерами. **Актер (actor)** представляет собой некую роль, которую пользователь играет по отношению к системе. Актерами могут быть пользователь, торговый представитель пользователя, менеджер по продажам и товаровед.

Актеры действуют в рамках прецедентов. Один актер может выполнять несколько прецедентов; и наоборот, в соответствии с одним прецедентом могут действовать несколько актеров. Обычно клиентов много, поэтому роль клиента могут играть многие люди. К тому же один человек может играть несколько ролей, например менеджер по продажам, выполняющий роль торгового представителя клиента. Актер не обязательно должен быть человеком. Если система предоставляет некоторый сервис другой компьютерной системе, то другая система является ***актером.***

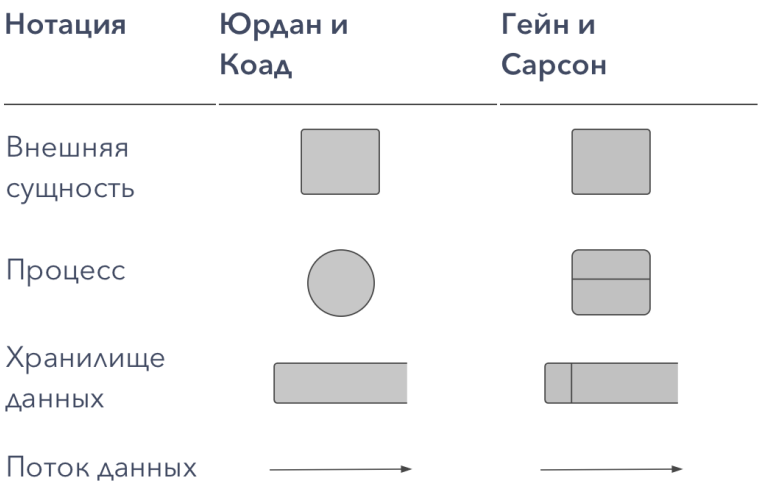
В каждом прецеденте есть ведущий актер, который посылает системе запрос на обслуживание. Ведущий актер – это актер, желание которого пытается удовлетворить прецедент и который обычно, но не всегда, является инициатором прецедента. Одновременно могут быть и другие актеры, с которыми система также взаимодействует во время выполнения прецедента. Они называются второстепенными актерами.

Каждый шаг в прецеденте – это элемент взаимодействия актера с системой. Каждый шаг должен быть простым утверждением и должен четко указывать, кто выполняет этот шаг. Шаг должен показывать намерение актера, а не механику его действий. Следовательно, в прецеденте интерфейс актера не описывается. Действительно, составление прецедента обычно предшествует разработке интерфейса пользователя.

DFD — общепринятое сокращение от англ. data flow diagrams — диаграммы потоков данных. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Диаграмма потоков данных (data flow diagram, DFD) — один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML.

DFD – это нотация, предназначенная для моделирования информационный систем с точки зрения хранения, обработки и передачи данных.

Исторически синтаксис этой нотации применяется в двух вариантах — Йордана (Yourdon) и Гейна-Сарсона (Gane-Sarson). Различия между ними – в таблице ниже:




Здесь можно использовать разные варианты, главное, чтобы они были понятны вам и вашим клиентам. Нотации DFD — удобный инструмент для создания нерегламентированных диаграмм, которые можно сделать быстро и с максимумом свободы.

Применяется этот вид нотации в случае, когда требуется описание системы как хранилища данных. Т.е. нотация должна наглядно ответить на вопросы:

Из чего состоит информационная система?

Что нужно, чтобы обработать информацию?

Непосредственно DFD нотация состоит из следующих элементов:

Процесс (англ. Process), т.е. функция или последовательность действий, которые нужно предпринять, чтобы данные были обработаны. Это может быть создание заказа, регистрация клиента и т.д. В названиях процессов принято использовать глаголы, т.е. «Создать клиента» (а не «создание клиента») или «обработать заказ» (а не «проведение заказа»). Здесь нет строгой системы требований, как, например, в IDEF0 или BPMN, где нотации имеют жестко определенный синтаксис, так как они могут быть исполняемыми. Но все же определенных правил стоит придерживаться, чтобы не вносить путаницу при чтении DFD другими людьми.

Внешние сущности (англ. External Entity). Это любые объекты, которые не входят в саму систему, но являются для нее источником информации либо получателями какой-либо информации из системы после обработки данных. Это может быть человек, внешняя система, какие-либо носители информации и хранилища данных.

Хранилище данных (англ. Data store). Внутреннее хранилище данных для процессов в системе. Поступившие данные перед обработкой и результат после обработки, а также промежуточные значения должны где-то храниться. Это и есть базы данных, таблицы или любой другой вариант организации и хранения данных. Здесь будут храниться данные о клиентах, заявки клиентов, расходные накладные и любые другие данные, которые поступили в систему или являются результатом обработки процессов.

Поток данных (англ. Data flow). В нотации отображается в виде стрелок, которые показывают, какая информация входит, а какая исходит из того или иного блока на диаграмме.

Нотация DFD может описывать любые действия, в том числе, процесс продажи или отгрузки товара, работу с заявками от клиентов или закупки материалов, с точки зрения описания системы. Эта нотация помогает понять, из чего должна состоять система, что нужно для автоматизации бизнес-процесса. Но DFD не является описанием непосредственно бизнес-процесса. Здесь, например, нет такого важного параметра, как время. Также в этой нотации не предусмотрены условия и «развилки». В DFD мы рассматриваем откуда появляются данные, какие данные нужны, их обработку и куда результаты отправить. Т.е. в этой нотации описывается не столько непосредственно процесс, сколько движение потоков данных. Для работы с процессами я рекомендую использовать BPMN или IDEF3 (о ней я расскажу в другой раз).

Как создавать нотации DFD

Давайте для примера рассмотрим нотацию автоматизации продаж. Допустим, у нас есть клиент, который делает заявку через сайт или по телефону. Есть менеджер, который регистрирует эту заявку. Таким образом, в системе появляются данные – клиент и его заказ. Работник склада должен это увидеть и произвести отгрузку товара с оформлением всех необходимых документов и передать документы клиенту.

Последовательность получается такая:

Клиент предоставляет свои данные и заявку.

Менеджер проверяет и вносит полученные данные в систему.

Работник склада формирует документы, например, расходную накладную, и отгружает товар.

Клиент получает товар и пакет документов к нему.

Эту последовательность действий нам необходимо увидеть с точки зрения хранения данных и работы с ними в IT-системе.

С точки зрения DFD у нас имеются:

Покупатель – это внешняя сущность, которая является источником данных и получением результата.

Процесс обработки заказа (подтверждение и проводка данных в системе менеджером).

Сбор заказа на складе (после получения заявки).

Оформление отгрузки (создание необходимых документов).

Какие правила необходимо знать, чтобы создать DFD диаграмму:

Каждый процесс должен иметь хотя бы один вход и один выход. Смысл процессов здесь заключается в обработке данных, а потому процесс должен получить данные (входящая стрелка) и отдать куда-то после обработки (исходящая стрелка);

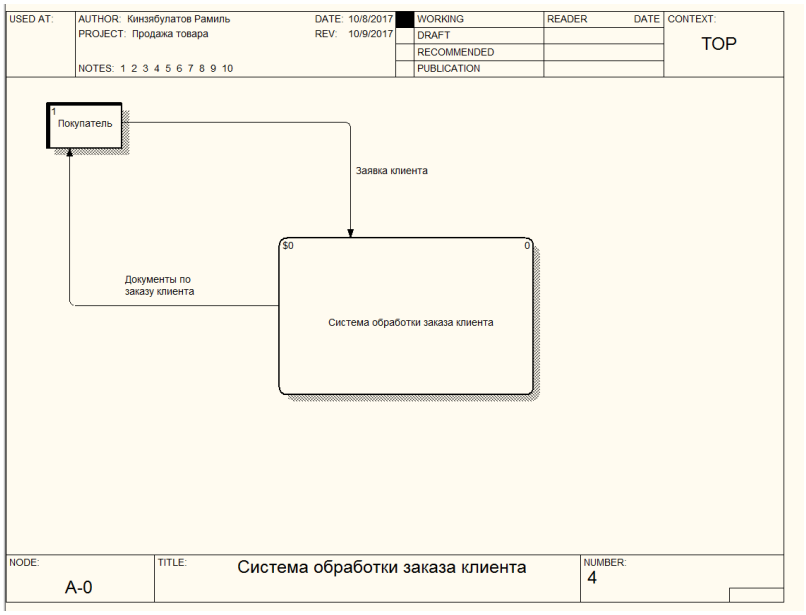
Процесс обработки данных должен иметь внешнюю входящую стрелку (данные от внешней сущности). Для того, чтобы любой подобный процесс начал работать, мало использовать данные из хранилища, должна поступить новая информация для последующей обработки;

Стрелки не могут связывать напрямую хранилища данных, все связи идут через процессы. Нет смысла просто перемещать данные из одного места в другое, а именно так читается прямая связь двух хранилищ стрелкой. Данные поступают для того, чтобы производились какие-то действия, в нашем примере – осуществлялся процесс продажи. А это возможно только посредством обработки (процесса);

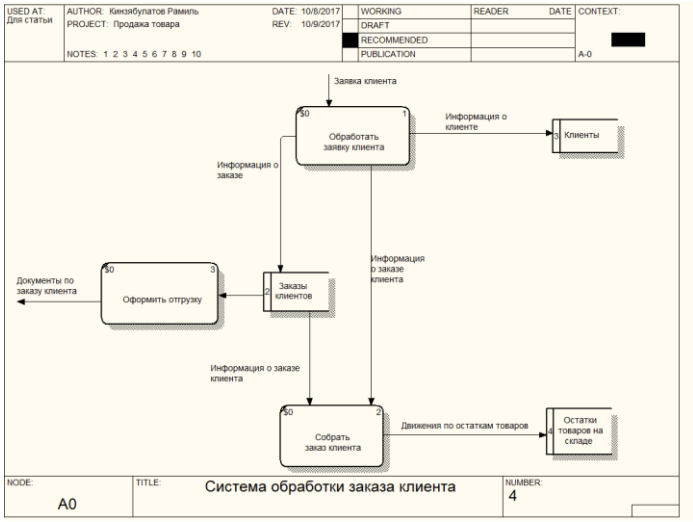
Все процессы должны быть связаны либо с другими процессами, либо с другими хранилищами данных. Процессы не существуют сами по себе, а потому результат должен куда-то передаваться;

Декомпозиция. В DFD-диаграммах предусмотрена возможность создавать крупные процессы и декомпозировать их на подпроцессы с подробным описанием действий. Например, мы можем создать процесс «создание заявки», который потом декомпозировать на последовательность действий, например, на получение заявки, отдельно – проверку и получение данных клиента, если товар в интернет-магазине продается под заказ, то также при формировании заявки потребуется получить данные от поставщика о наличии нужных наименований и т.д. И тогда на верхней диаграмме у нас будет блок «обработка заявки», а при декомпозировании мы получим диаграмму с подробной последовательностью действий на этом этапе. При этом ни на одном этапе у нас не будет условий и ветвления. Будет процесс и его декомпозиция глубиной до 3-4 уровней.

Как будет выглядеть диаграмма (без декомпозиции, верхний уровень):




И декомпозиция основного элемента нашей диаграммы:




Где используются DFD нотации

DFD-диаграммы активно применяются при разработке программного обеспечения. При этом:

хранилища данных – это электронные таблицы и базы данных,

внешние сущности – клиенты или другие базы данных, в том числе, из других программ (интеграция и обмен данными),

процессы – это выполняемые функции и модули в системе.

Также DFD нотации удобны при анализе, когда система рассматривается с точки зрения документооборота. При этом можно наглядно увидеть, где хранятся данные, каким образом производится обмен документацией, где в этом процессе допущены ошибки организации бизнес-процессов и пр. Но здесь применение DFD диаграмм требует особой осторожности. Все же это не описание бизнес-процесса как такового, а, скорее, диаграмма перемещения данных при реализации бизнес-процессов. Но как вспомогательный вариант, в том числе, для наглядной демонстрации клиенту существующих проблем и методов оптимизации работы, этот вид нотаций вполне подойдет.

Например, для выявления проблем документооборота, дублирования документов или, наоборот, недостающей документации или электронных данных в системе, очень удобно создать отдельно – описание бизнес-процесса, а потом к нему – DFD-нотацию. Либо наоборот, предварительно для понимания основ работы бизнеса и особенностей реализации документооборота создается DFD-нотация. Она помогает выявить, например, отсутствие в системе автоматизации важных документов, которые на самом деле создаются (на бумаге), но в системе никак не отображаются. А потом уже строится оптимизированный бизнес-процесс с учетом выявленных нюансов документооборота.

DFD нотации – это просто!

Главное, четко понимать ограничения построения этого типа диаграмм (отсутствие условий, времени и т.д.) и применять их там, где именно такой подход окажется удобнее.

Что в DFD-нотациях особенно удобно, здесь не обязательно придерживаться строгих правил и синтаксиса. Эти нотации не будут исполнимыми, они нужны для понимания особенностей документооборота, структуры и последующей работы с данными. А потому, если ваша диаграмма понятна и вам, и заказчику, какие-то отступления от стандартов DFD вполне допустимы.

Примеры диаграмм: