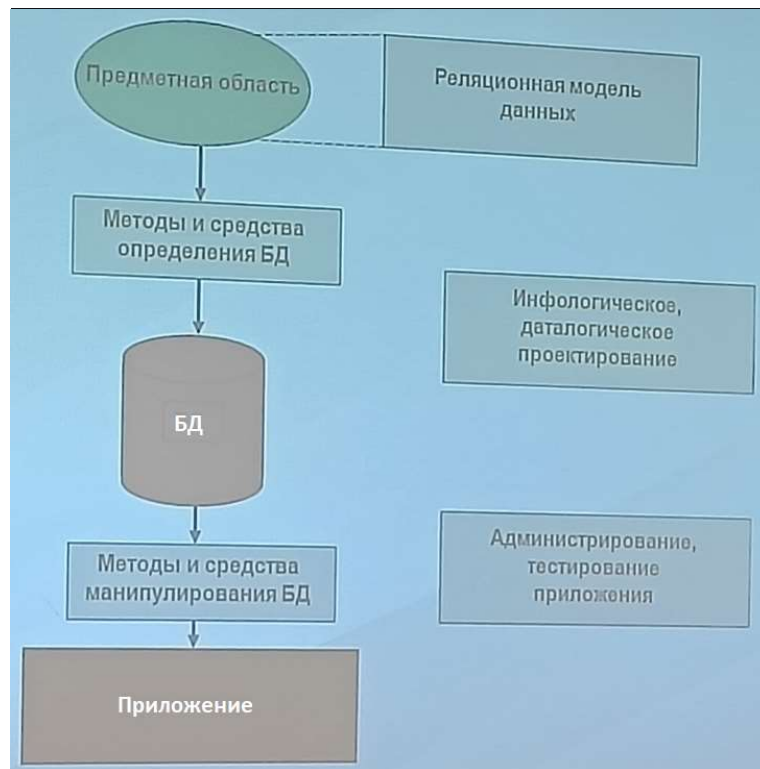


Оглавление

Проектирование информационных систем	2
Жизненный цикл информационной системы.....	2
Модели жизненного цикла ПО ИС.....	3
Каскадно-возвратная модель.....	4
Спиральная модель	6
Технологии проектирования информационных систем.....	6
Методологии проектирования информационных систем	8
Проблемы	10
Структурный подход к проектированию ИС.....	10
Обобщенная схема уровней моделирования базы данных	13

Проектирование информационных систем



Проектирование ИС ранее велось на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС.

Жизненный цикл информационной системы

Жизненный цикл информационной системы - это базовое понятие методологии проектирования ИС, это непрерывный процесс, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент полного изъятия из эксплуатации.

Документ, регламентирующий жизненный цикл - международный стандарт ISO/IEC 12207.

Структура жизненного цикла базируется на 3 группах процессов:

1. Основные процессы:

- Приобретение
- Поставка
- Разработка
- Сопровождение

2. Вспомогательные процессы:

- Документирование
- Управление конфигурацией
- Обеспечение качества
- Верификация (проверка состояния ИС на этом этапе)
- Аттестация
- Оценка
- Аудит
- Решение проблем

3. Организационные процессы:

- Управление проектом
- Создание инфраструктуры проекта
- Определение, оценка и улучшение самого жизненного цикла
- Обучение

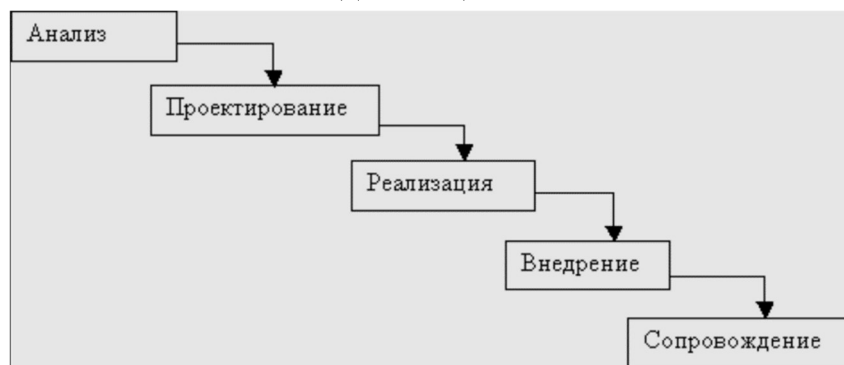
Каждый процесс характеризуется задачами, методами их решения, исходными данными, полученными на предыдущем этапе и результатами.

Модели жизненного цикла ПО ИС

Известны 2 основные модели жизненного цикла

Каскадная модель

Каскадная модель (1970-1985 гг.) - раньше приложение представляло собой единое целое.



Плюсы:

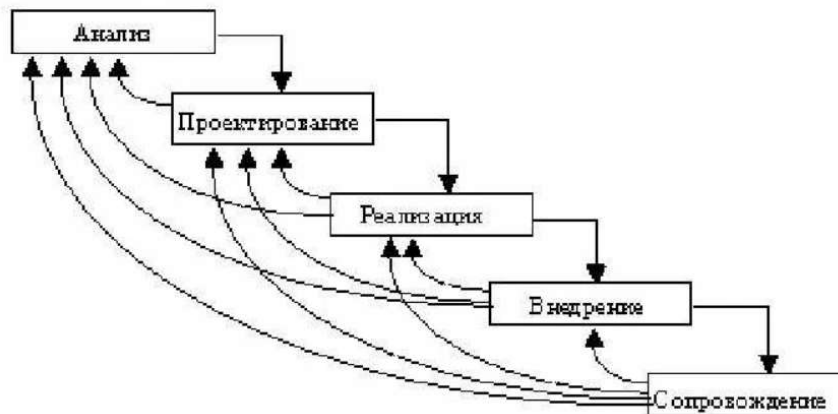
- Выполняемые в логической последовательности этапы позволяют планировать сроки и затраты

- Каскадная модель используется при построении ИС, для которых точно и полно сформулированы все требования
- На каждом этапе формируется законченный набор документации

Минусы:

- Реальный процесс создания ИС не укладывается в эту жесткую схему

Каскадно-возвратная модель



1. Анализ системы и объектоуправление:

- Обследуем систему, изучаем систему управления
- Анализ существующей организационной структуры, применяемых технологий производства, системы документооборота, связей с внешними организациями, системами
- Создаются модель системы и объект управления, которые предназначены для выявления, анализа недостатков существующей системы
- Моделируется деятельность организации, проводится бизнес-инжиниринг важнейших функций
- Формулируются требования к создаваемой ИС, методам и технологиям работ, инструментальным средствам ИС, разрабатывается план создания
- Создается ТЗ

2. Проектирование:

- Разрабатывается концепция ИС, создаются информационная и организационная структуры управления, разрабатывается архитектура ИС

- Проектируется структура БД, производится нормализация, выполняется конфигурирование вычислительной компьютерной сети
- Для приложений определяется требование к информационным технологиям, разрабатываются алгоритм обработки данных, формализованные описания решения задач, осуществляется выбор программных средств базового и прикладного назначения

3. Реализация (программирование, разработка, кодирование) - это этап технической реализации программных решений

- Создание и проектирование форм документов, заполнение классификаторов и кодификаторов технико-экономической информацией, программная реализация приложения, создание проектной документации
- По мере разработки отдельных программных компонентов осуществляется их тестирование, интеграция, для пользователей разрабатывается эксплуатационная документация

4. Внедрение - занимает от нескольких месяцев до нескольких лет

- Осуществляется первоначальная загрузка нормативно-справочной информации, ввод в схему документооборота новых форм документов
- Внедрение разбивается на опытную и промышленную стадии эксплуатации ИС. Промышленная стадия начинается после приемки

5. Сопровождение - наиболее длительный этап. В процессе эксплуатации регистрируются ошибки, проводится экспертиза проектных решений, формулируются требования к модификации в связи с изменениями объекта и функций управления, появлением новых информационных технологий

Основной недостаток каскадного подхода - запаздывание в получении результатов. Согласование результатов с пользователем производится только в точках, планируемых после завершения каждого этапа. Требования к ИС заморожены в виде ТЗ на все время ее создания. Эти проблемы решены в спиральной модели.

Спиральная модель



1. Определение требований
2. Анализ
3. Проектирование
4. Реализация, тестирование
5. Интеграция

Каждый виток спирали соответствует созданию фрагмента или версии ПО. На нем уточняются цели и характеристики проекта, определяются его качества и планируется работа следующего витка. Разработка итерациями отражает объективно существующий спиральный цикл создания системы. Неполное завершение работ на каждом этапе позволяет переходить на следующий, не дожидаясь полного завершения работ на текущем. Недостающую работу можно выполнить на следующей итерации. Главная задача - быстрее показать пользователю системы рабочий продукт, этим оптимизируя процесс уточнения, дополнения требований. Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения вводят временное ограничение на каждый из этапов ЖЦ. Переход осуществляется в соответствии с планом, даже если работа не закончена. План составляется на основе статистических данных, полученных из предыдущих проектов и личного опыта.

Технологии проектирования информационных систем

В основе проекта любой ИС лежат методологии, технологии и инструментальные средства проектирования. Методология реализуется через технологии с привлечением инструментальных средств.

Технологии проектирования - совокупность 3 составляющих:

1. Технологические инструкции, состоящие из описания последовательности технологических операций
2. Условия выполнения этих операций
3. Описание этих операций в виде графических и текстовых нотаций

Технология проектирования, разработки и сопровождения ИС должна удовлетворять следующим требованиям:

- Поддерживает полный ЖЦ
- Достигает цели разработки с заданным качеством в установленное время
- Крупные проекты должны разбиваться на подсистемы, слабо связанные по данным и функциям
- Работа ведется небольшими группами
- Обеспечивает минимальное время получения работоспособной информационной системы, но не всей, а отдельных подсистем. Практика показывает, что при наличии завершенного проекта внедрение идет по отдельным подсистемам
- Возможность ведения версий проекта, автоматического выпуска проектной документации
- Проектные решения должны быть независимы от средств реализации операционных систем, СУБД, языков программирования
- Должна строиться на согласованных инструментальных средствах проектирования

Реальное применение любой технологии проектирования невозможно без выработки стандартов, т. е. правил, соглашений:

- **Стандарт проектирования** - устанавливает набор необходимых моделей и степень их детализации
- **Стандарт оформления проектной документации** - правила фиксации проектных решений на диаграммах, правила именования объектов, набор атрибутов для объектов, правила оформления диаграмм, форма, размер объектов, конфигурация рабочих мест разработчиков. Состав, структура документации на каждой стадии проектирования. Требования к оформлению - содержание раздела, пунктов, таблиц. Правило подготовки, рассмотрения, согласования, утверждения документов с указанием предельных сроков. Требование к настройке издательской системы для подготовки документации. Требования к настройке case-средств

- **Стандарт пользовательского интерфейса** - правило оформления экранов: шрифты, цвета, состав, расположение окон и элементов управления. Правило использования клавиатуры и мыши. Правило оформления текста помощи. Перечень стандартных сообщений. Правило обработки реакции пользователя

Методологии проектирования информационных систем

RAD (Rapid Application Development) - методология быстрой разработки приложений. Реализует спиральную модель ЖЦ. Процесс разработки содержит 3 принципа:

1. Небольшая команда разработчиков (2-10)
2. Короткий, но тщательно проработанный производственный график (2-6 месяцев)
3. Повторяющийся цикл, при котором разработчики по мере развития приложения запрашивают и реализуют в продукте новые требования, полученное со стороны заказчика

ЖЦ по этой методологии состоит из 4 фаз:

- **1 фаза - фаза анализа и планирования требований:**
 - Пользователи определяют функции системы, выделяют приоритетные
 - Уточняются требования со стороны заказчика
 - Ограничивается масштаб проекта
 - Определяются временные рамки для фаз
 - Определяется сама возможность реализации проекта в рамках финансирования на данных аппаратных средствах.

Результат:

- Список и приоритетность ИС
- Предварительный функционал
- **2 фаза - фаза проектирования:**
 - Пользователь принимает участие в проектировании системы
 - Уточняются требования к системе

- Более подробно рассматриваются процессы системы
- Анализируется и корректируется функциональная модель
- При необходимости для каждого процесса создается частичный прототип: экран, диалог, отчет
- Определяются требования доступа к данным
- Определяется набор необходимой документации
- Оценивается количество функциональных элементов
- Принимается решение разделения ИС на подсистемы, для реализации которой требуется 2-3 месяца

Результат:

- Общая информационная модель системы
- Функциональные модели системы в целом и подсистем, определенные с помощью case-средств
- Интерфейсы между автономно разрабатываемыми подсистемами
- Построены прототипы экранов, отчетов, диалогов

• **3 фаза - фаза построения:**

- Быстрая разработка приложения
- Программный код генерируется автоматически с помощью case-средств
- Конечные пользователи оценивают результаты, вносят корректировки, осуществляют тестирование
- Завершается физическое проектирование системы
- Определяются необходимые распределения данных
- Проводится анализ использования
- Завершается физическое проектирование БД
- Уточняются и фиксируются требования к аппаратуре, определяются способы увеличения производительности
- Оформляется документация

Результат: готовая система

• **4 фаза - фаза внедрения:**

- Обучение пользователей
- Работа с существующей системой

Проблемы

Методология RAD не универсальна, хороша для небольших проектов, для конкретного заказчика. Неприменима для типовых и сложных программ, построения операционных систем, программ требующих большого объема уникального кода (операционная система), приложений, в которых отсутствует ярко выраженная интерфейсная часть - приложения реального времени, приложения от которых зависит безопасность людей.

Основные принципы методологии RAD:

1. Разработка приложений итеративно
2. Неполное завершение работ на каждом этапе
3. Вовлечение пользователей в процесс разработки
4. Применение case средств
5. Использование генераторов кода
6. Построение прототипов
7. Тестирование и развитие одновременно с разработкой
8. Ведение разработки малой хорошо управляемой командой
9. Грамотное руководство, четкое планирование, контроль

Структурный подход к проектированию ИС

Суть: ИС разбиваются на информационные подсистемы, которые делятся на подсистемы, подзадачи, вплоть до мелких процедур. Существующие методологии структурного программирования делятся на подпринципы:

1. **Разделяй и властвуй:** сложная проблема решается путем ее разбиения на множество меньших независимых задач
2. **Иерархическое упорядочивание:** составные части организуются в иерархические древовидные структуры с добавлением новых деталей на каждом уровне
3. **Абстрагирование:** выделение существенных аспектов системы и отвлечение от несущественных

4. **Принцип формализации:** строгий методический подход к решению проблемы
5. **Принцип непротиворечивости:** элементы системы должны быть обоснованы и согласованы
6. **Принцип структурирования данных:** данные должны быть структурированы и иерархически организованы

Существуют 3 методологии структурного подхода:

1. **SADT** (методология структурного анализа и моделирования)
2. **DFD** (диаграммы потоков данных)
3. **ERD** (диаграммы сущность-связь)

Нотация была введена Ченом, а развитие получила от Баркера.

Шаги при моделировании ERD:

1. Происходит выделение сущности:
 - i. Сущность - это реальный или воображаемый объект, имеющий значение для рассматривания предметной области, информация которой подлежит хранению. Каждая сущность должна обладать уникальным идентификатором, свойствами, иметь уникальное имя. К одному и тому же имени применяется одна и та же интерпретация
 - ii. Обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр
 - iii. Обладает одним или несколькими атрибутами, которые принадлежат сущности или наследуются через связь
 - iv. Сущность может обладать любым количеством связей с другими сущностями модели
2. Идентификация связей. Связь - это поименованная ассоциация между двумя сущностями. Каждый экземпляр одной сущности (родительская) ассоциирован с произвольным количеством экземпляром второй сущности (потомок). Связь имеет имя, выраженное грамматическим оборотом глагола, и помещенное рядом с линией связи. Имя связи формулируется с точки зрения родителя. Предложение о связи образовано соединением имени сущности родителя, имени связи, выражения степени, имени сущности потомка.
3. Идентификация атрибутов: любая характеристика сущности, предназначенная для идентификации, квалификации, классификации,

количественной характеристики или выражения состояния сущности.

Атрибут может быть обязательным или необязательным. Обязательность означает, что атрибут не может принимать неопределенные значения (null).

Атрибут может быть либо описательным (дескриптор), либо быть уникальным идентификатором (первичным ключом).

Обобщенная схема уровней моделирования базы данных

Обобщенная схема уровней моделирования базы данных:

Уровень	Что описывает	Чем оперирует
Инфологический (концептуальный уровень)	Предметная область без привязки к виду баз данных	Сущности, атрибуты, некоторые связи
Даталогический (логический) уровень	Предметная область с привязкой к виду базы данных или даже конкретной СУБД	Сущности, атрибуты, связи, ключи, некоторые индексы и представления
Физический уровень	Технические аспекты реализации базы данных под управлением конкретной СУБД	Сущности, атрибуты, связи, ключи, индексы, представления, триггеры, хранимые подпрограммы, методы доступа, кодировки, права доступа и т. д. и т. п.

С продвижением вниз по таблице увеличивается степень детализации.

Инфологический + даталогический = логический уровень.

Инфологический (концептуальный) уровень моделирования ставит своей целью создание т. н. концептуальной модели, отражающей основные сущности предметной области, их атрибуты и связи (возможно, пока не все) между сущностями.

Упрощенно: описание предметов и явлений реального мира, данные о которых потом будут помещены в базу данных.

Как предметную область целиком, так и непосредственно данные на этом уровне можно описывать различными моделями (например, семантической, графовой, "сущность-связь"). В качестве способа представления модели чаще всего используется UML или простое словесное описание (что особенно удобно для обсуждения модели с представителем заказчика, не являющимся IT-специалистом).

Даталогический уровень (часто его называют просто "логическим") моделирования детализирует инфологическую модель, превращая ее в логическую схему, на которой ранее выявленные сущности, атрибуты и связи оформляются согласно правилам моделирования для выбранного типа БД (возможно, даже с учетом конкретной СУБД).

Упрощенно: описание предметов и явлений реального мира по правилам выбранной СУБД.

Во многих средствах проектирования баз данных, поддерживающих разделение лишь на "логическое" и "физическое" проектирование именно этот уровень называется "логическим".

Здесь уже появляются вполне привычные таблицы, поля, ключи, связи, индексы и представления - все то, что и составляет суть базы данных. Поскольку реализация этих объектов зависит от вида базы данных (и даже конкретной СУБД), здесь уже нельзя строить строго абстрактную модель, и приходится учитывать типичные особенности выбранных базы данных и СУБД.

Даталогических моделей много (документальная, фактографическая, теоретико-графовая, теоретико-множественная, объектно-ориентированная, основанная на инвертированных файлах и т. д.). И каждая из них породила свой вид баз данных.

Физический уровень моделирования продолжает детализацию и позволяет создавать т. н. физическую схему, на которой максимально учитываются технические особенности работы конкретной СУБД и ее возможности по организации и управлению структурами разрабатываемой базы данных и данными в ней.

На этом уровне модель данных может быть представлена так же, как и на предыдущем (дatalogическом) - чаще всего, в виде UML, но одной лишь графической формы здесь недостаточно, потому в ход идут SQL-скрипты, словесные описания необходимых изменений и настроек, фрагменты конфигурационных файлов и т. д.

Цели и задачи проектирования на инфологическом уровне

Цель - создание модели, отражающей сущности предметной области, их атрибуты и связи (возможно, пока не все) между сущностями.

Т. е. необходимо максимально глубоко исследовать предметную область и выразить ее понятия в виде сущностей, атрибутов, связей.

Исследование предметной области как раз и представляет основную сложность, разделяющуюся на локальные проблемы:

- извлечение информации
- глубина исследования
- границы исследования

Извлечение информации - универсальная проблема для любого проекта (даже вне контекста баз данных). Как правило, у заказчика нет готового полного технического описания проекта, потому необходимо организовать сбор требований к проекту, и в них выделить те требования, которые прямо или косвенно относятся к базе данных.

Техники выявления требований:

- Интервью
- Работа с фокусными группами
- Анкетирование
- Семинары и мозговой штурм
- Наблюдение
- Прототипирование
- Анализ документов
- Моделирование процессов и взаимодействий
- и т. д. и т. п.

Глубина исследования. Недостаточно выявить лишь некоторые сущности и некоторые их атрибуты. Необходимо выявить их все.

Например, база данных реальной библиотеки не может состоять из таблиц "книги" (название, год выпуска) и "авторы" (ФИО, год рождения) - в такой базе данных будут десятки таблиц с десятками полей.

Проблема усложняется в силу того, что в общем случае - это не задача заказчика - продумывать все возможные детали, нюансы, аспекты. Это задача именно проектировщика базы данных.

Границы исследования - проблема часто не техническая, а управленческая: в силу настояния заказчика или из технических соображений в базу данных могут начать попадать сущности, вполне реально существующие и связанные с предметной областью, но неактуальные для реализуемого проекта.

Так, (пример с библиотекой) можно от самой базы данных библиотеки перейти к логистике (книги как-то в библиотеку попадают), складскому учету (книги где-то хранятся), бухгалтерии (ведь финансовые вопросы надо как-то решать), кадровому учету (ведь в библиотеке работают сотрудники) и т. д.

Обеспечение качества БД

Как **данные**, так и **структура** базы данных должны обеспечивать выполнение принципов:

- **Полнота.** Частая ошибка - неполный анализ предметной области и, как следствие, отсутствие в базе данных той или иной информации. Например, описывая сотрудника некоей организации указывают лишь ФИО и дату рождения, но забывают про адрес проживания, телефоны (их несколько), адреса электронной почты (несколько), паспортные данные, сведения об образовании...

- **Уникальность.** В контексте требований **нормализации** это "требование неизбыточности данных". В БД в двух и более местах хранятся одни и те же данные
- **Учет изменений во времени.** Некоторые данные требуется обрабатывать с учетом привязки ко времени. К этой же проблеме относится несвоевременное обновление кэширующих таблиц и полей и в целом выполнение требования актуальности данных
- **Корректность (валидность).** Дилемма: быть ли БД гибкой или следовать стандартам и ограничениям. В случае жестких ограничений мы упрощаем обработку данных, но ставим в тупик пользователей, чьи реальные данные не соответствуют нашим правилам.

Решения:

- Подсказки при вводе
 - Выбор предложенного
 - Возможность добавить свой вариант
- **Точность.** С какой точностью указывать рост или вес пользователя? С какой точностью хранить цену товара? С какой точностью фиксировать время наступления события? "Чем точнее, тем лучше"?
- **Единообразие представления.** Бессмысленное разнообразие нарушает требование удобства использования базы данных

Как обеспечить качество БД

- **Сбор, анализ и обсуждение информации.** Источники информации: заказчик, пользователи, разработчики работающих с базой данных приложений, объективно существующие ограничения предметной области и т. д.
- **Выбор технических решений.** Выбор вида СУБД, конкретной версии и реализации этой СУБД, сопутствующей инфраструктуры - все это может накладывать отпечаток на модель создаваемой базы данных
- **Контроль и самоконтроль.** Даже в идеальных условиях остается вероятность что-то перепутать, забыть, неправильно учесть и неверно отразить в модели базы данных. Важно многократно перепроверять результаты своей работы, получать обратную связь от коллег-специалистов и от всех заинтересованных сторон
- **Эксперименты.** Начиная с даталогического уровня проектирования можно периодически экспортировать модель БД в СУБД, наполнять ее тестовыми

данными и исследовать ее поведение. Это позволяет увидеть проблемы с производительностью и иные сложнопредсказуемые сложности, покажет ряд ошибок, появившихся в силу невнимательности и легко отслеживаемых средствами самой СУБД

- **"А что, если?" vs "Ура, работает!"**. Ошибка начинающих - прекращение анализа проблемы после того, как найдено первое же работающее решение. Но, во-первых, это решение может не быть самым лучшим (и дальнейший анализ позволил бы найти лучший вариант), а во-вторых, такое первое попавшееся решение может покрывать лишь частные случаи, и в других ситуациях (с другими данными, другими условиями выполнения, другим способом взаимодействия с БД и т. д.) приводить к возникновению ошибок. Потому поиск нескольких альтернативных решений, выбор наилучшего из них и анализ выбранного решения в различных ситуациях ощутимо повышает качество разрабатываемой БД
- **Тщательное документирование**
- **Расширение собственного кругозора**