Создание модели бизнеса

- Описание новой компании – это работа по прямому инжинирингу, которая начинается с формулирования целей и образа (vision) будущей компании. После этого набрасываются различные сценарии. Для каждого сценария создается общее описание процесса включающее заказчиков, поставщиков и т. д., а также сам процесс. Далее проводится имитационное моделирование различных процессов – при помощи деловой игры или компьютерной модели. Наконец, выбранная альтернатива реализуется.

Традиционные способы разработки моделей

1. **Структурный анализ н структурное проектирование (Structured Analysis and Structured Design – SA/SD)** - система предоставляет своим пользователям одну или несколько функций – подход функциональной декомпозиции. SA/SD предлагает набор средств, таких, как диаграммы потоков данных, диаграммы состояний- переходов, ER-диаграммы (на фазе анализа) и структурные схемы (на фазе проектирования).
2. **Методика IDEF (Integrated computer aided manufacturing DEFinition)** была разработана ВВС США на основе идей, появившихся в середине 70-х гг. На основе этой методики Министерство обороны США создало Федеральный стандарт обработки информации IDEF1X, который обеспечивает поддержку на нескольких уровнях посредством «модели бизнеса», «модели информационной системы» И модели технологии». Моделирование бизнеса поддерживается ER-диаграммами для данных и диаграммами потоков данных специального вида, что позволяет иерархически описывать функции системы.
3. **Методика SADT (Structured Analysis and Design Technique)** использует систему обозначений, похожую на диаграммы потоков данных в IDEF, для описания функций и структур данных информационной системы на основе декомпозиции.

- При описании информационной системы предполагается, что она содержит два типа сущностей: некоторый аналог программы (операционные сущности, которые выполняют некоторую обработку) и данные (пассивные сущности, которые хранят информацию, доступную для поиска, чтения и замены). Другими словами, информационная система описывается как некая абстракция компьютера.
- При моделировании (разработке) сложные информационные системы разбиваются на составные части, каждая из которых рассматривается отдельно от других (декомпозиция). Классический подход к разработке сложных систем представляет собой структурное проектирование, при котором осуществляется алгоритмическая декомпозиция системы по методу «сверху-вниз».

- Жизненный цикл разработки сложной системы в этом случае складывается из этапов анализа, проектирования, программирования, тестирования и сопровождения, которые выполняются последовательно. Такой метод, называемый **каскадным**, имеет следующие отличительные особенности:

1. Линейность выполнения этапов жизненного цикла разработки;
2. Четкое разделение данных и процессов их обработки;
3. Использование процедурных языков программирования.

Главный недостаток: последовательное выполнение этапов.

Для устранения этого недостатка был предложен ***спиральный подход.*** Он заключается в том, что разработка проекта ведется как бы по спирали, причем на каждом ее витке выполняются последовательно перечисленные выше этапы, на которых уточняется проект. Этот подход дополняет каскадный метод элементами итеративности.

Недостатки спирального подхода:

- трудоемкость внесения изменений;
- большой объем документации по проекту, затрудняющий программирование;
- серьезные ограничения возможностей сборки системы из готовых компонентов;
- сложность переноса на другие платформы.

Объектно-ориентированный подход к разработке моделей

Особенности сложных информационных систем

1. Иерархичность

Иерархические структуры позволяют рассматривать только определенный уровень, не вдаваясь в детали реализации. Для сложной системы целесообразно моделировать два типа иерархии – типовую и структурную. Типовая иерархия отражает взаимосвязи «общее – частное». В объектно –ориентированном подходе ей соответствует иерархия классов. Структурная иерархия показывает связи типа «это-часть того». При объектно-ориентированном подходе ей соответствует иерархия объектов.

2. Групповая разработка

Иерархический характер сложных систем согласуется с принципом групповой разработки. Деятельность каждого участника проекта ограничивается соответствующим иерархическим уровнем. Применяемые инструментальные средства (ИС) должны поддерживать групповую разработку. Для этого ИС реализуются на комплексах с архитектурой «клиент-сервер». В них должна быть предусмотрена возможность интеграции результатов работы отдельных участников проекта и защиты от несанкционированного доступа.

3. Модифицируемость проекта

Сложные системы обычно подвергаются многократной модификации. Это связано как с устранением ошибок, выявленных в процессе разработки, отладки и эксплуатации, так и с необходимостью внесения изменений и дополнений, вызванных изменениями внешних условий и требований к системе. Это может вызвать определенные трудности, ввиду значительного объема таких систем и большим числом взаимосвязей между их компонентами.

4. Сборочное проектирование

При разработке больших информационных систем широко используется концепция сборочного проектирования, основанная на идее повторно используемых компонентов. Сборка прикладной системы из готовых компонентов позволяет значительно сократить время разработки. Определяющее значение имеет насколько применяемые методики и поддерживающие их ИС обладают возможностями создания повторно используемых компонентов.

5. Использование стандартных СУБД

Интеграция прикладной системы с базой данных ставит перед разработчиками задачу обеспечения преемственности, т.е. возможности использования в разрабатываемом приложении данных, накопленных в БД. Кроме того, при разработке приложения возникает необходимость проектирования логической структуры новой БД. Для интегрированных систем с клиент-серверной архитектурой используются специальные инструментальные средства.

Особенности объектно-ориентированного подхода (ООП)

В основе ООП лежат понятия объект, класс, инкапсуляция, наследование и полиморфизм. В качестве объекта могут рассматриваться конкретные предметы, а также абстрактные или реальные сущности. Объектами могут быть покупатель, фирма, банк, заказ на поставку. Объект обладает индивидуальностью или поведением, имеет атрибуты, значения которых определяют его состояние. Пример: конкретный покупатель , делая заказ, может оказаться в состоянии, когда денег на его счете не хватает для оплаты, а его «поведение» в этом случае заключается в «обращении в банк за кредитом».

Каждый объект является представителем некоторого класса однотипных объектов. Класс определяет общие свойства для всех его объектов. К таким свойствам относятся:

1. Состав и структура данных, описывающих атрибуты класса и соответствующих объектов;
2. Совокупность методов –процедур, определяющих взаимодействие объектов этого класса с внешней средой.

Например, описание класса «магазины» может включать некоторые атрибуты (индивидуальные для каждого объекта этого класса – конкретного магазина): «название», «адрес», «штат сотрудников», «текущий счет», а также методы: «формирование заказов на поставку товаров»; «передача товара со склада в торговую секцию». Объекты и классы обладают характерными свойствами, которые активно используются при объектно-ориентированном подходе и во многом определяют его преимущества.

Инкапсуляция – скрытие информации. При ООП предусмотрена возможность запретить любой доступ к атрибутам объектов, кроме как через его методы. Внутренняя структура объекта в этом случае скрыта для пользователя, т.е. объекты можно считать самостоятельными сущностями, отделенными от внешнего мира. Для того, чтобы объект произвел некоторое действие, ему извне необходимо послать сообщение, которое инициирует выполнение нужного метода. Инкапсуляция позволяет изменять реализацию любого класса объектов без опасения, что это вызовет нежелательные побочные эффекты в программной системе. Тем самым упрощается процесс исправления ошибок и модификации программ.

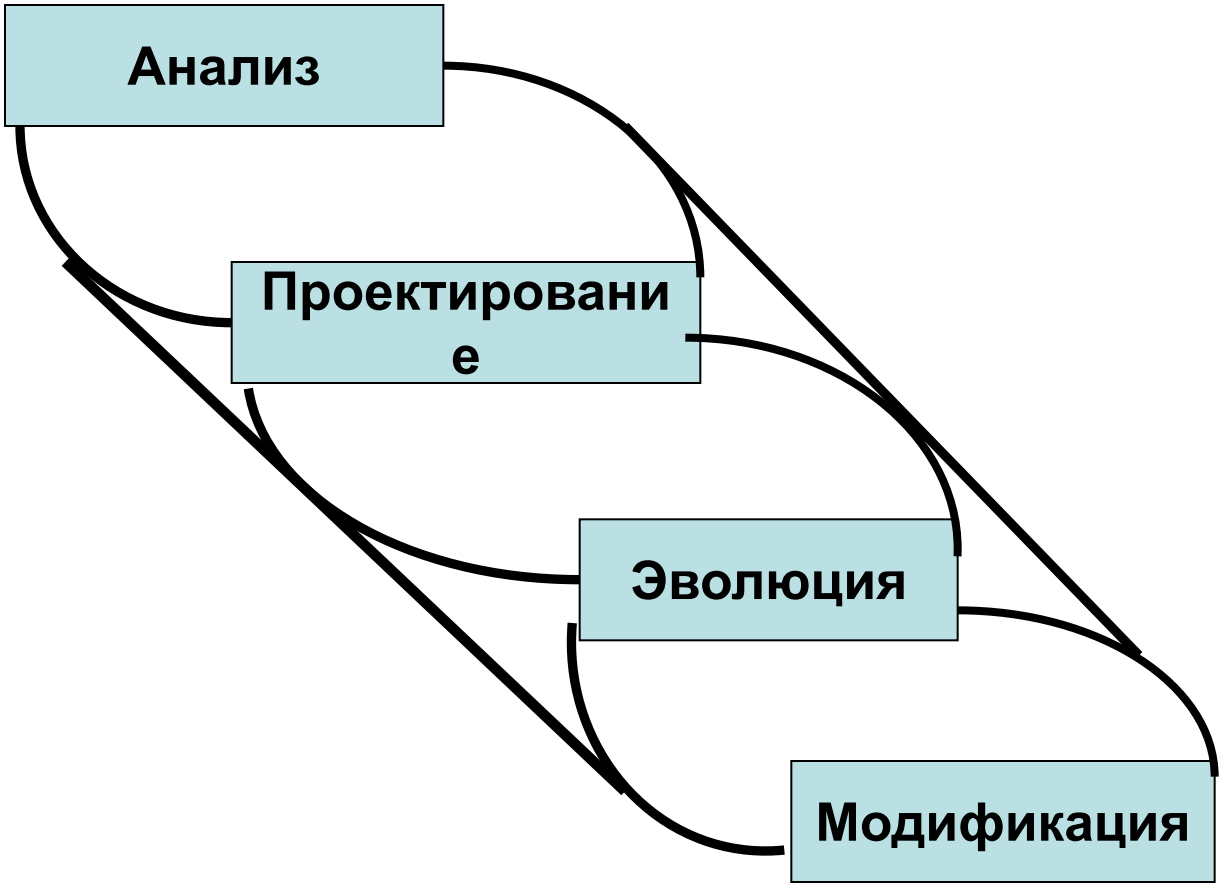
Наследование – возможность создавать из классов новые классы по принципу «от общего к частному».

Наследование позволяет новым классам при сохранении всех свойств классов-родителей (суперклассы) добавлять свои черты, отражающие х индивидуальность. Новый класс должен содержать только коды и данные для новых или изменяющихся методов. Сообщения, обработка которых не обеспечивается собственными методами класса, передаются суперклассу. Наследование позволяет создавать иерархии классов и является эффективным средством внесения изменений и дополнений в программные системы.

Полиморфизм — способность объектов выбирать метод на основе типов данных, принимаемых в сообщении. Каждый объект может реагировать по своему на одно и то же сообщение. Полиморфизм позволяет упростить исходные тексты программ, обеспечивает их развитие за счет введения новых методов обработки.

Объектно-ориентированная декомпозиция заключается в представлении системы в виде совокупности классов и объектов предметной области. При этом иерархический характер сложной системы отражается в виде иерархии классов, а ее функционирование рассматривается как взаимодействие объектов.

Жизненный цикл объектно-ориентированной разработки программных систем содержит несколько этапов, но в отличие от структурного подхода в нем нет строгой последовательности их выполнения. Процесс носит принципиально итеративный характер, что полностью отвечает потребностям разработчиков.



Цикл разработки сложных систем с использованием объектно-ориентированного подхода

Цикл разработки приложения при использовании объектно-ориентированного подхода

Разработка: на этапе обследования – объектно-ориентированного анализа, определяются требования к системе. При осуществлении анализа предметной области определяются основные классы и объекты. Результатом обследования должны быть достаточно полные сведения для создания модели системы.

Проектирование: при объектно-ориентированном проектировании детализируется представление классов и объектов, полученных на этапе анализа. Определяются структуры данных, методы, отношения между классами, разрабатываются сценарии взаимодействия объектов. При проектировании системы могут вводиться новые классы и объекты, если это потребуется для решения поставленных проблем. Выходом процесса проектирования является детальная модель системы, составлены спецификации объектов, классов и отношений, достаточные для их программирования.

Эволюция системы: этап, включающий в себя программирование, тестирование и сборку системы. ООП обеспечивает быстрое создание прототипов проектируемой системы, постепенное развитие которых приводит к конечному результату. На этом этапе возможно введение новых классов, изменение структур данных, добавление новых методов. Программирование и тестирование отдельных компонентов системы возможно до завершения проектирования, что экономит время разработки.

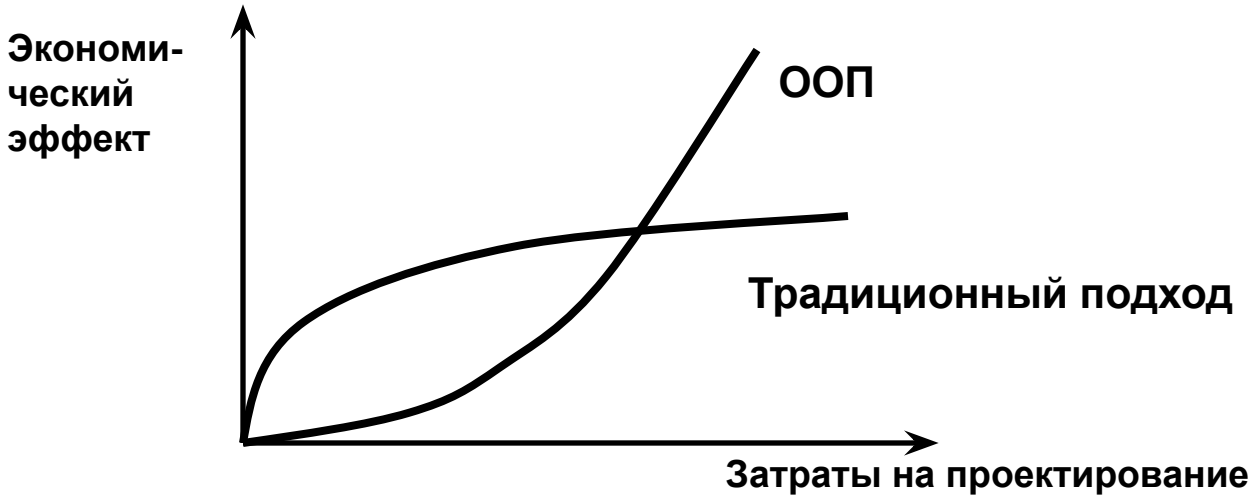
Модификация: при ООП существует возможность внесения изменений, при этом не требуется полного пересмотра проекта, затрагиваются лишь необходимые для этого классы и объекты.

Преимущества ООП

- Распараллеливание работ (программирование и тестирование отдельных компонентов системы возможно до завершения проектирования, что экономит время разработки. При программировании может возникнуть необходимость внесения изменений в существующие классы или потребоваться введение новых объектов или классов. Тогда, вернувшись к этапу проектирования или даже к анализу, можно внести изменения и дополнения, не подвергая объект полной переработке.)
- Упрощение внесения изменений (в ООП внесение изменений в проект имеет локальный характер. В тех случаях, когда изменение носит характер уточнения, детализации, вводят новые классы, наследующие поведение ранее созданных. Наследование – одно из основных свойств классов – позволяет в этих случаях не только не пересматривать ранее созданные объекты и классы, но даже обойтись без их повторной трансляции. В более сложных случаях, когда меняются методы, определяющие интерфейс классов, изменения в проекте будут более значительными, но и тогда они будут локализованы, затрагивая лишь классы, использующие эти методы.)
- Гибкая архитектура и переносимость (объектно-ориентированная декомпозиция, в результате которой приложение представляется в виде совокупности классов и объектов, обеспечивает гибкость архитектуры системы. В клиент - сервисной системе объекты могут размещаться как на местах клиента, так и на сервере. В гетерогенных (разнородных) сетях возможна реализация классов на компьютерах разных типов, а фиксированный интерфейс каждого класса, определяемый наботом его методов, обеспечит правильность функционирования системы. Изменения конфигурации оборудования не требуют внесения изменений в проект).

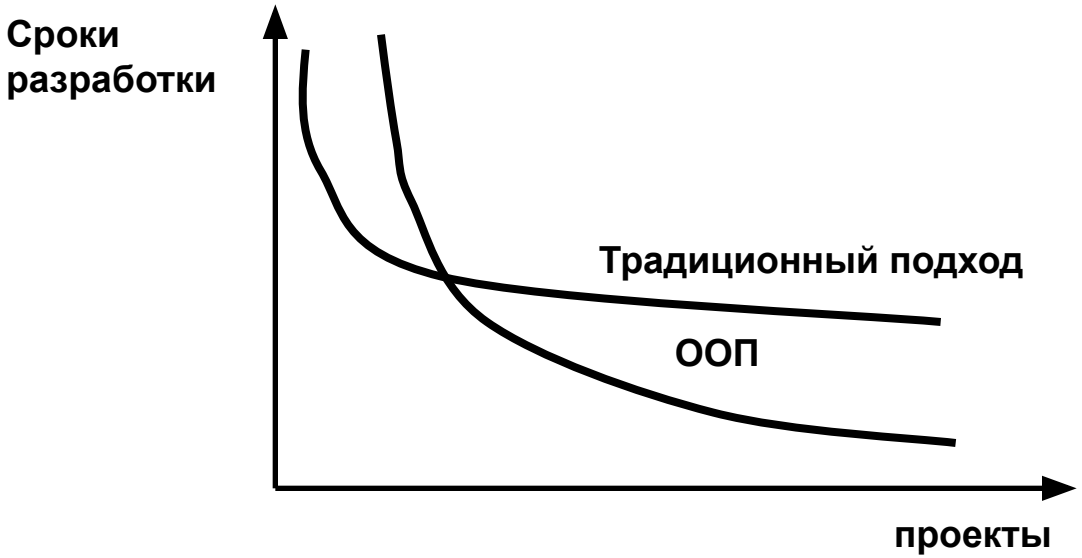
- Повторное использование программных компонентов (разрабатываемые в рамках некоторого приложения классы обычно отражают типовые решения, поэтому их использование возможно и в других приложениях. Возможность повторного использования программных компонентов – одна из самых сильных и привлекательных черт ООП. Библиотеки классов, отражающие опыт в определенной области, позволяют значительно снизить объем программирования при разработке новых приложений. При наличии развитых библиотек классов проектирование и программирование новых приложений будет в основном сводиться к сборке системы из готовых компонентов. Иерархический характер сложных программных систем позволяет значительно повысить эффективность повторного использования компонентов. Основные свойства классов и объектов – инкапсуляция, наследование и полиморфизм – полностью отвечают задаче повторного использования).
- Естественность описания (ООП позволяет описывать как статические, так и динамические отношения между объектами модели. По описанию предметной области, выполненному на естественном языке, легко выделить объекты и статические связи между ними. Объекты соответствуют существительным, а связи – глаголам и отглагольным формам).

Объектно-ориентированные технологии не дают немедленной отдачи. Эффект от их применения начинает сказываться после разработки двух-трех проектов и накопления повторно используемых компонентов, отражающих типовые проектные решения в данной области.



Рост эффективности разработок по отношению к затратам при традиционном и ООП

При ООП кривая эффективности резко взмывает вверх благодаря сборке систем из готовых программных компонентов



Снижение сроков разработки при традиционном и ООП

Для традиционного подхода снижение времени разработки связано в основном с ростом квалификации участников проектов, для ООП прибавляется опыт использования типовых проектных решений

Интегрированные подходы к разработке моделей

Объектно-ориентированное моделирование

-базовая методология БПР.

CASE- технологии

-ориентация на разработчиков ИС привела к тому, что их начинают объединять с другими современными технологиями (например, с ООП)

Имитационное моделирование

-обеспечивает:

- наиболее глубокое представление моделей для непрограммирующего пользователя
- наиболее полные средства анализа таких моделей