CASE- технологии

CASE (Computer Aided Software Engeneering)

Эти технологии являются естественным продолжением эволюции всей отрасли разработки ПО.

CASE-1: анализ требований, проектирование спецификаций и структуры, редактирование интерфейсов.

CASE-2: генерация исходных текстов и реализация интегрированного окружения поддержки полного ЖЦ разработки ПО.

Начальный толчок создания CASE - **структурное программирование**

1) достаточно развитый уровень формализации.

1) Необходимость автоматизировать выполнение рутинных работ.

В настоящее время CASE-технологии используются не только для производства ПО, но и как мощный инструмент решения исследовательских и проектных задач.

Попытка смоделировать систему вообще.

Назначение CASE для помощи в создании ПО:

- автоматизация процесса построения ПО;
- обеспечение функций реверсивного проектирования;
- обеспечение функций сопровождения ПО.
- поддержка разработки моделей анализа и проектирования ПО;

Истоки CASE - использование разработчиками различных схем и рисунков

Систематизация графических средств приводит к созданию “графических” языков, которые могут рассматриваться как языки более высокого уровня по отношению к текстам программ.

Основные задачи CASE-систем

1. **Разработка моделей предметной области, функциональной структуры системы, структур данных на графических языках.**
2. **Хранение моделей в единой базе данных – репозитории, доступном всем участникам разработки.**
3. **Формальный анализ разрабатываемых моделей, позволяющий избегать некоторых семантических ошибок.**
4. **Автоматизированная генерация структур баз данных, приложений, текстов программ.**
5. **Автоматизированная генерация документации на программные системы.**
6. **Обеспечение повторного использования наработок при модернизации, перепроектировании системы.**

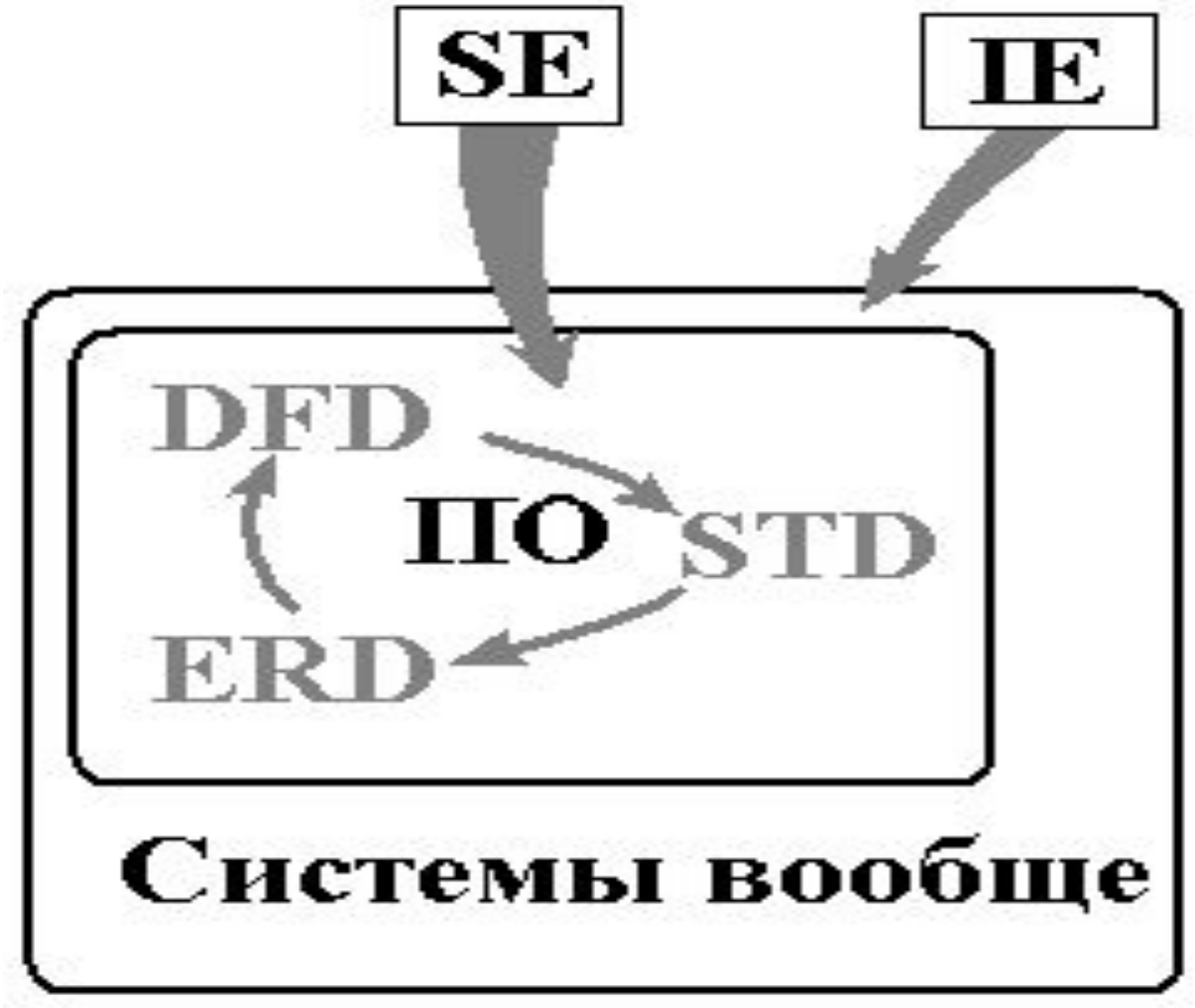
CASE-визуальное средство для **структурного анализа**:

- **DFD** (***Data Flow Diagrams***) - диаграммы потоков данных;
- **ERD** (***Entity-Relationship Diagrams***) - диаграммы ‘сущность - связь’;
- **STD** (***State Transition Diagrams***) - диаграммы переходов состояний.

методологии структурного анализа классифицируются по признакам:

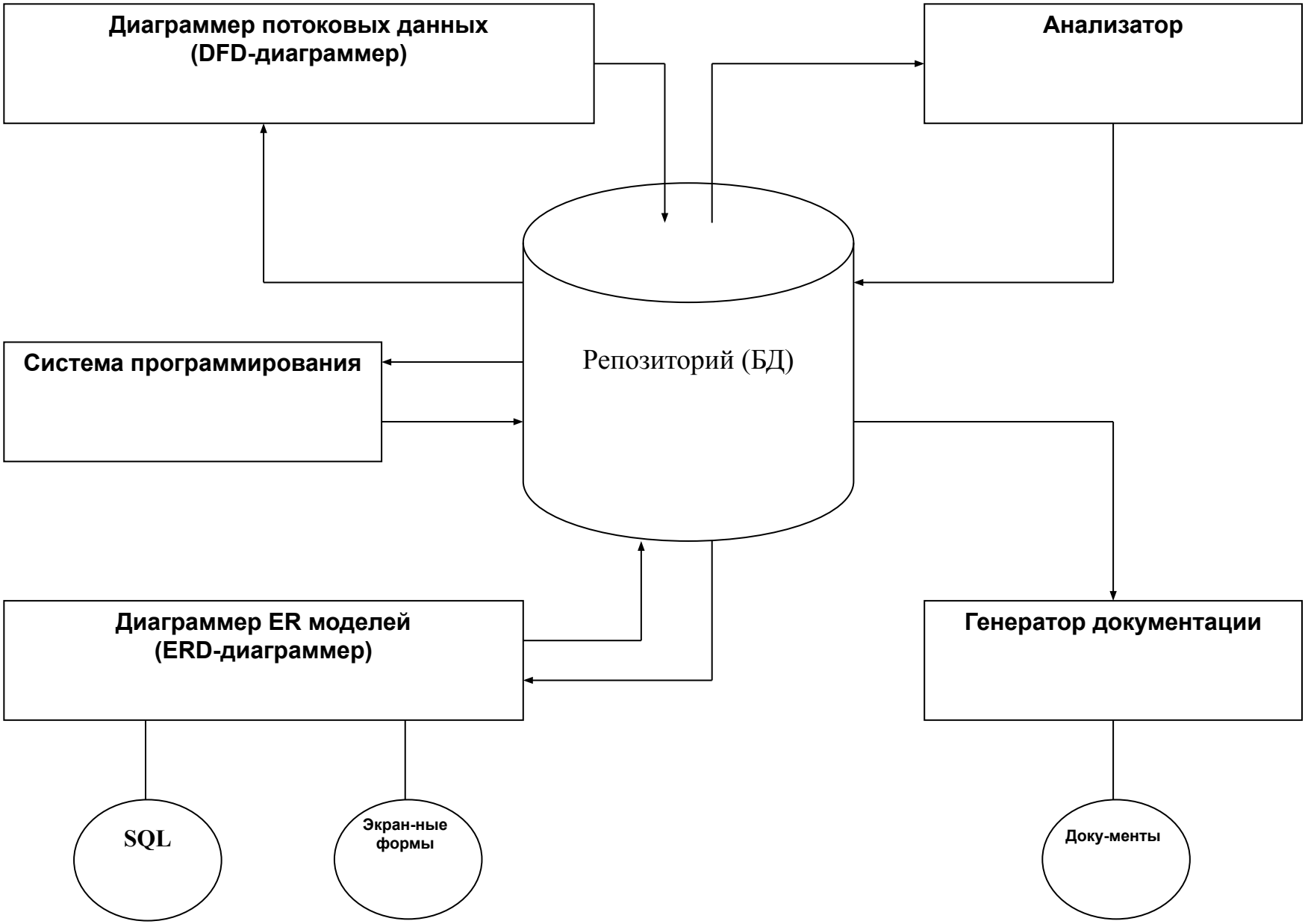
- по отношению к школам - ***Software Engineering*** (**SE**) и ***Information Engineering*** (**IE**);
- по порядку построения моделей - процедурно-ориентированные, ориентированные на данные и информационно-ориентированные;
- по типу целевых систем - для систем реального времени и для информационных систем.

SE - нисходящий подход;
IE - более новая дисциплина.



Состав типовой CASE-системы

- диаграммеры,
- средства для конструирования пользовательского интерфейса,
- генераторы приложений,
- генераторы документации,
- система программирования,
- центральная база данных проекта – репозиторий


Диаграммер потоковых данных
(DFD-диаграммер)
Анализатор
Репозиторий (БД)
Система программирования
Диаграммер ER моделей
(ERD-диаграммер)
Генератор документации
SQL
Экран-ные
формы
Доку-менты

Изменение распределения трудозатрат

Технология	**Этапы разработки**			
	Анализ	Проектирование	Кодирование	Тестирование
Традиционная	20%	15%	20%	45%
CASE-I	30%	30%	15%	25%
CASE-II	40%	40%	5%	15%

Основные CASE-средства:

ERWIN (разработка ER-моделей),
BPWIN (разработка диаграмм потоков данных),
POWER DESIGNER,
DESIGNER 2000,
RATIONAL ROSE,
PARADIGM+

Классификация по функциональной ориентации

Анализ и проектирование.

CASE- аналитик (Эйтекс);

- POSE (Computer Systems Advisers);
- Design/IDEF (Meta Software);
- BPWin (Logic Works);
- SELECT (Select Software Tools);
- CASE/4/0 (micro TOOl GmbH)

Проектирование баз данных и файлов.

- ERWin (Logic Works);
- S-Designor (SPD);
- Designtr/2000 (Oracle);
- Sillverrun (Computer Systems Advisers)/

Программирование.

- COBOL 2/Workbench (Mikro Focus);
- DECASE (DEC);
- NETRON/CAP (Netron);
- APS (Sage Softwfre).

Сопровождение и реинжениринг

- Adpac CASE Tools (Adpac);
- Scan/COBOL и SuperStructure (Computer Data Systems):
- Inshtctor/Recoder (language Tecnologe).

CASE-средства фирмы **Computer Associated**

- **AllFusion Process Modeler** (ранее:BPwin) - моделирование бизнес-процессов
- **AllFusion ERwin Data Modeler** (ранее: ERwin) - моделирование данных
- **AllFusion Data Model Validator** (ранее: ERwin Examiner) - проверка моделей данных.
- **AllFusion Model Manager** (ранее: ModelMart) - сервер для совместной работы пользователей ERwin и/или Bpwin
- **AllFusion Saphir Option** - – средство просмотра структур данных широкого набора корпоративных информационных систем.
- **AllFusion Component Modeler** (Paradigm Plus) - моделирование компонентов ПО

Альтернативой структурному подходу стали объектно-ориентированные методы разработки ИС. В первой половине 90-х годов был предложен универсальный язык объектного проектирования - ***Unified Modeling Language***, UML (The Unified Method, Draft Edition (0.8). Rational Software Corporation, October 1995).

- Существует несколько CASE-средств, поддерживающих язык UML. Наиболее известными являются:
- CASE-средства, поддерживающие UML:
- **Paradigm Plus** фирмы PLATINUM technology (Computer Associated).
- **Rational Rose** фирмы Rational Software.
- SELECT фирмы SELECT Software

RAD
(Rapid Application Development)

методология быстрой разработки приложений

- небольшая команда программистов (от 2 до 10 человек);
- короткий, производственный график (от 2 до 6 мес);
- итерационный подход, через взаимодействие с заказчиком.

ЖЦ ПО по методологии RAD состоит из четырех фаз:

- анализа и планирования требований;
- проектирования;
- реализации;
- внедрения.

На фазе анализа:

- определение требований силами пользователей под руководством специалистов-разработчиков.
- Определяется возможность реализации проекта в установленных рамках финансирования, на данных аппаратных средствах и т. п.
- Определяются временные рамки самого проекта в каждой из последующих фаз.

На фазе проектирования:

- Пользователи, взаимодействуя с разработчиками, уточняют и дополняют требования Для быстрого получения работающих прототипов приложений **используются CASE-средства.**
- Анализируется и корректируется функциональная модель. Каждый процесс рассматривается детально, создается частичный прототип: экран, диалог, отчет и пр. Принимается решение о количестве, составляющих ПО подсистем, поддающихся разработке одной командой.

Результат данной фазы:

- общая информационная модель системы;
- функциональные модели системы в целом и подсистем, реализуемых отдельными командами;
- точно определенные с помощью CASE-средства интерфейсы между автономно разрабатываемыми подсистемами;
- построенные прототипы экранов, отчетов, диалогов.

На фазе реализации:

- Программный код частично формируется с помощью автоматических генераторов CASE-средств.
- Для контроля за выполнением требований к ПО привлекаются конечные пользователи.
- Во время разработки осуществляется тестирование каждой подсистемы.
- Разрабатываемые подсистемы постепенно внедряются в общую систему. Производится их тестирование и тестирование всей системы в целом.
- Завершается физическое проектирование системы. Если необходимо, создаются базы данных, завершается разработка документации ПО.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

Принципы организации RAD:

- **Обязательное использование инструментальных средств.**
- **Тесное взаимодействие между разработчиками и заказчиком**.
- **Работа ведется немногочисленными хорошо управляемыми группами профессионалов.**
- Разработка базируется на моделях.
- **Итерационное прототипирование (традиционно 3 прототипа).**
- **RAD группа всегда работает только над одним прототипом.**
- **Большие системы разбиваются на подсистемы и** для него выделяется несколько RAD групп.

Гибкое проектирование и XP

Гибкое моделирование (Adile Modeling - AM) – это упорядочивающая, основанная на практическом опыте методология эффективного моделирования и документирования программных систем.

Манифест альянса гибкой разработки ПО

(февраль 2001 года)

1. Люди и контакты важнее процессов и средств.
2. Работающие программы важнее идеальной документации.
3. Сотрудничество с заказчиком важнее переговоров по условиям контракта.
4. Готовность к изменениям важнее соблюдения планов.

Принципы гибкой разработки ПО:

1. Мы придаем первоочередное значение удовлетворению заказчика, быстро и постоянно предоставляя нужное ему программное обеспечение.
2. Мы приветствуем изменения требований даже на поздних этапах разработки. Гибкие процессы позволяют поддерживать изменения, обеспечивая заказчику конкурентное преимущество.
3. Новые версии работающего ПО поставляются часто, с регулярностью от нескольких недель до нескольких месяцев, причем более предпочтительны короткие временные периоды.
4. В ходе проекта бизнесмены и разработчики должны постоянно работать вместе.
5. Проекты строятся мотивированными индивидуалами. Создайте им условия, удовлетворяйте их требования и доверяйте им в том, что касается выполнения работы.
6. Наиболее производительный и эффективный способ передачи информации рабочей группе и внутри нее – это разговор лицом к лицу.
7. Работающее ПО – это основной показатель прогресса.
8. Гибкие процессы стимулируют устойчивую работу. Спонсоры, разработчики и пользователи должны быть в состоянии неограниченно долго поддерживать постоянный ритм работы.
9. Непрерывное внимание к техническому качеству и хорошему проектированию улучшает гибкость.
10. Простота – искусство минимизировать количество ненужной работы – исключительно важна.
11. Наилучшим образом архитектура, требования и проектирование формируются и выполняются самоорганизующимися командами.
12. Команда должна регулярно обсуждать, как повысить эффективность своей работы, после чего изменять и согласовывать рабочий процесс с результатами этих обсуждений.

Экстремальное программирование (XP)

Ghbywbgs:

- Ищите самое простое решение, которое может сработать.
- Это вам не понадобится (не делать ничеговпрок).
- программный код должен быть максимально прост:
  - Система успешно проходит все тесты;
  - Код системы ясно раскрывает все изначальные замыслы;
  - В ней отсутствует дублирование кода;
  - Используется минимально возможное количество классов и методов