

## 12. Средства автоматизации проектирования

Для автоматизации проектирования и разработки информационных систем в 70–80-е гг. широко применялась *структурная* методология, означающая использование формализованных методов описания разрабатываемой системы и принимаемых технических решений. При этом использовались графические средства описания различных моделей информационных систем с помощью схем и диаграмм. При ручной разработке информационных систем такие графические модели разрабатывать и использовать весьма трудоемко.

Отмеченные обстоятельства послужили одной из причин появления программно-технологических средств, получивших название CASE–средств и реализующих CASE-технологии создания и сопровождения информационных систем. Кроме структурной методологии, в ряде современных CASE–средств используется объектно-ориентированная методология проектирования.

Термин **CASE** (*Computer Aided Software Engineering*) дословно переводится как разработка программного обеспечения с помощью компьютера. В настоящее время этот термин получил более широкий смысл, означающий автоматизацию разработки информационных систем.

**CASE–средства** представляют собой программные средства, поддерживающие процессы создания и/или сопровождения информационных систем, такие как: анализ и формулировка требований, проектирование баз данных и приложений, генерация кода, тестирование, обеспечение качества, управление конфигурацией и проектом.

**CASE–систему** можно определить как набор CASE–средств, имеющих определенное функциональное предназначение и выполненных в рамках единого программного продукта.

Модель жизненного цикла (ЖЦ) программного обеспечения информационной системы (ПО ИС) при автоматизированном проектировании ИС играет достаточно важную роль. Это обусловлено тем, что каждая из CASE–систем поддерживает определенную модель ЖЦ ПО ИС.

**Жизненный цикл ПО ИС** представляет собой непрерывный процесс, начинающийся с момента принятия решения о создании ПО и заканчивающийся при завершении его эксплуатации.

Основным нормативным документом, регламентирующим ЖЦ ПО, является международный стандарт ISO/IEC 12207 (International Organization of Standardization — Международная организация по стандартизации / International Electrotechnical Commission — международная комиссия по электротехнике). В нем определяется структура ЖЦ, содержащая процессы, действия и задачи, которые должны быть выполнены при создании ПО.

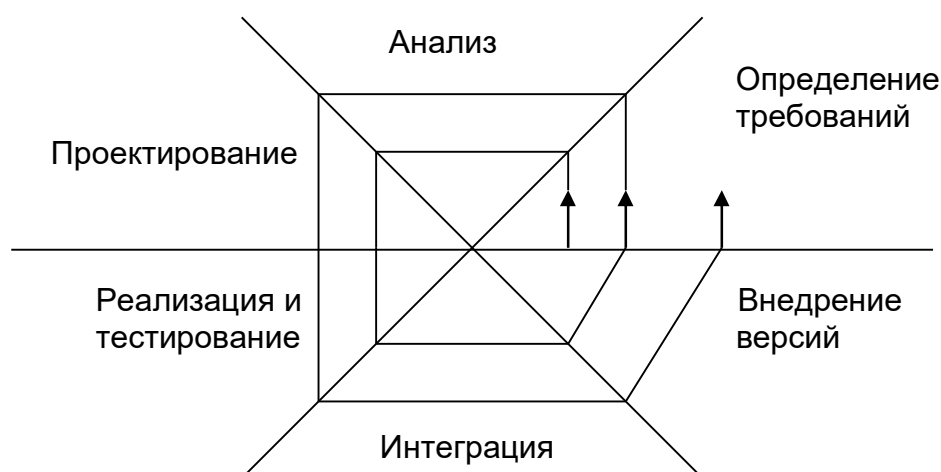
Под **моделью ЖЦ** ПО понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении ЖЦ. Наибольшее распространение получили следующие модели ЖЦ ПО: каскадная, с промежуточным контролем и спиральная.

Модели каскадная и с промежуточным контролем включают следующие этапы жизненного цикла ПО: анализ, проектирование, реализация, внедрение и сопровождение.

**Каскадная модель** предполагает строго последовательную реализацию перечисленных этапов жизненного цикла. Достоинствами такой модели являются: формирование на каждом этапе законченного комплекта документации и возможность планирования сроков завершения работ и соответствующих затрат. Недостатком модели является ее несоответствие реальному процессу создания ПО, который обычно не укладывается в жесткую схему и требует возврата к предыдущим этапам для уточнения или пересмотра принятых решений.

**Модель с промежуточным контролем** приближает жизненный цикл к реальному процессу создания и применения ПО. В отличие от каскадной модели, она допускает возврат с каждого этапа жизненного цикла на любой предыдущий этап для выполнения межэтапной корректировки. При этом обеспечивается большая надежность ПО, но вместе с тем увеличивается длительность периода разработки.

**Спиральная модель** жизненного цикла (рисунок 1) позволяет устранить недостатки предыдущих моделей. Основной упор в ней делается на начальные этапы: анализ и проектирование. На них реализуемость технических решений проверяется с помощью создания прототипов.



**Рисунок 1** — Спиральная модель жизненного цикла

При спиральной схеме разработки неполное завершение работ на очередном этапе позволяет переходить на следующий этап. Незавершенная работа может выполняться на следующем витке спирали. Тем самым обеспечивается возможность предъявить пользователям системы ее некоторый работоспособный вариант для уточнения требований.

При **классификации CASE–средств** используют следующие признаки:

- ориентацию на этапы жизненного цикла;
- функциональную полноту;
- тип используемой модели;
- степень независимости от СУБД;
- допустимые платформы.

Рассмотрим классификацию CASE–средств по наиболее часто используемым признакам.

По **ориентации** на этапы жизненного цикла выделяют следующие основные типы CASE–средств:

- *средства анализа*, предназначенные для построения и анализа моделей предметной области, например: Design/IDEF (Meta Software) и BPwin (Logic Works);
- *средства анализа и проектирования*, обеспечивающие создание проектных спецификаций, например. Vantage Team Builder (Cayenne), Silverrun (Silverrun Technologies), PRO-IV (McDonnell Douglas) и CASE.Аналитик (МакроПроджект),
- *средства проектирования баз данных*, обеспечивающие моделирование данных и разработку схем баз данных для основных СУБД, например: ERwin (Logic Works), S-Designor (SPD), DataBase Designer (ORACLE);
- *средства разработки приложений*, например: Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Centura) и Delphi (Borland).

По **функциональной полноте** CASE–системы и средства можно условно разделить на следующие типы:

- системы, предназначенные для *решения частных задач* на одном или нескольких этапах жизненного цикла, например, ERwin (Logic Works), S-Designor (SPD), CASE.Аналитик (МакроПроджект) и Silverrun (Silverrun Technologies);

- *интегрированные системы*, поддерживающие весь жизненный цикл ИС и связанные с общим репозиторием, например, система Vantage Team Builder (Cayenne) и система Designer/2000 с системой разработки приложений Developer/2000 (ORACLE).

По *типу используемых моделей* CASE–системы условно можно разделить на три основные разновидности: структурные, объектно-ориентированные и комбинированные.

Исторически первые *структурные* CASE–системы основаны на методах структурного и модульного программирования, структурного анализа и синтеза, например, система Vantage Team Builder (Cayenne).

*Объектно-ориентированные* методы и CASE–системы получили массовое использование с начала 90-х годов. Они позволяют сократить сроки разработки, а также повысить надежность и эффективность функционирования ИС. Примерами объектно-ориентированных CASE–систем являются Rational Rose (Rational Software) и Object Team (Cayenne).

*Комбинированные* инструментальные средства поддерживают одновременно структурные и объектно-ориентированные методы, например: Designer/2000 (ORACLE).

По *степени независимости от СУБД* CASE–системы можно разделить на две группы:

- независимые системы;
- встроенные в СУБД.

*Независимые* CASE–системы поставляются в виде автономных систем, не входящих в состав конкретной СУБД. Обычно они поддерживают несколько форматов баз данных через интерфейс ODBC. К числу независимых CASE–систем относятся S-Designor (SDP, Powersoft), ERwin (LogicWorks) и Silverrun (Computer Systems Advisers Inc.).

*Встроенные* CASE–системы обычно поддерживают главным образом формат баз данных СУБД, в состав которой они входят. При этом возможна поддержка и других форматов баз данных. Примером встроенной системы является Designer/2000, входящая в состав СУБД ORACLE.