Основные подходы к проектированию ПП

Каскадная схема разработки

Фиксированный порядок выполнения:

- Формирование требований;
- Анализ;
- Проектирование;
- Реализация;
- Тестирование;
- Внедрение;
- Эксплуатация и сопровождение

Требования, *определенные на 1-й стадии, строго выполняются во время разработки.*

Каждая стадия *завершается выпуском полного комплекта документации для последующих стадий.*

Достоинства:

- легко определить сроки и затраты на проект, т.е. **простота планирования** процесса разработки;

- получение в конце каждой стадии **законченного набора проектной документации**;

- **эффективна** при создании небольших систем.

Проблемы:

- предполагается полная корректность результатов каждого этапа (при разработке сложных систем это практически невозможно из-за отсутствия удовлетворительных средств описания на ранних стадиях);

- данная схема не предоставляет возможность пересмотра ранее принятых решений и изменения требований заказчика;

- не учитывается моральное старение используемых технических и программных средств.

Схема с промежуточным контролем

- Предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых похожа на «мини-проект».
- Цель каждой *итерации* — повысить функциональность разрабатываемой системы. Результат финальной итерации содержит всю требуемую функциональность продукта.
- Для успешного создания сложной системы нужно обеспечить необходимое количество небольших шагов с четкими критериями успеха, а также **возможность «отката»** к предыдущему успешному этапу в случае неудачи.

Такая схема носит итерационный характер. Основная опасность в том, что разработка не будет завершена в срок, постоянно находясь в состоянии уточнения и усовершенствования.

Спиральная схема

(Предложена в середине 80-х годов)

Основа - метод **прототипирования.**

Схема итерационная.

Каждая итерация **соответствует созданию:**

- фрагмента
- прототипа
- версии ПП

На каждой итерации:

- уточняются цели проекта;
- уточняются характеристики проекта;
- оценивается качество полученных результатов;
- планируются работы следующей итерации.

Метод прототипирования базируется на создании прототипов.

Прототипом называют действующий программный продукт, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного обеспечения.

На первой итерации:

- ✓ специфицируют
- ✓ проектируют
- ✓ реализуют
- ✓ тестируют

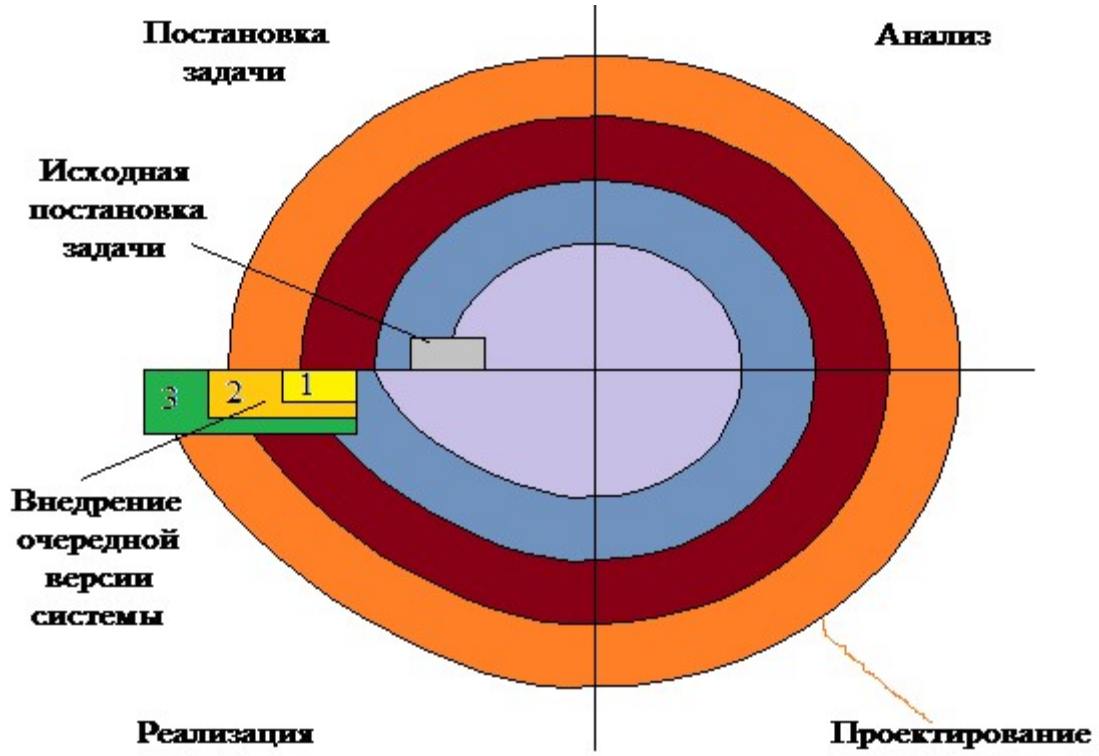
➢ Интерфейс

На второй:

- ✓ добавляют ограниченный набор функций.

На последующих:

- ✓ набор расширяют.



Основное достоинство: *при обеспечении определенной функциональной полноты, продукт можно предоставить (заказчику).*

Что это дает?

Это позволяет:

- сократить время до появления первых версий программного продукта;
- заинтересовать большое количество пользователей, а следовательно обеспечить быстрое продвижение на рынке;
- ускорить формирование и уточнение спецификаций за счет появления практики использования продукта;
- уменьшить вероятность морального устаревания системы за время разработки.

Основная проблема спиральной схемы:

- определение моментов перехода на следующие стадии.

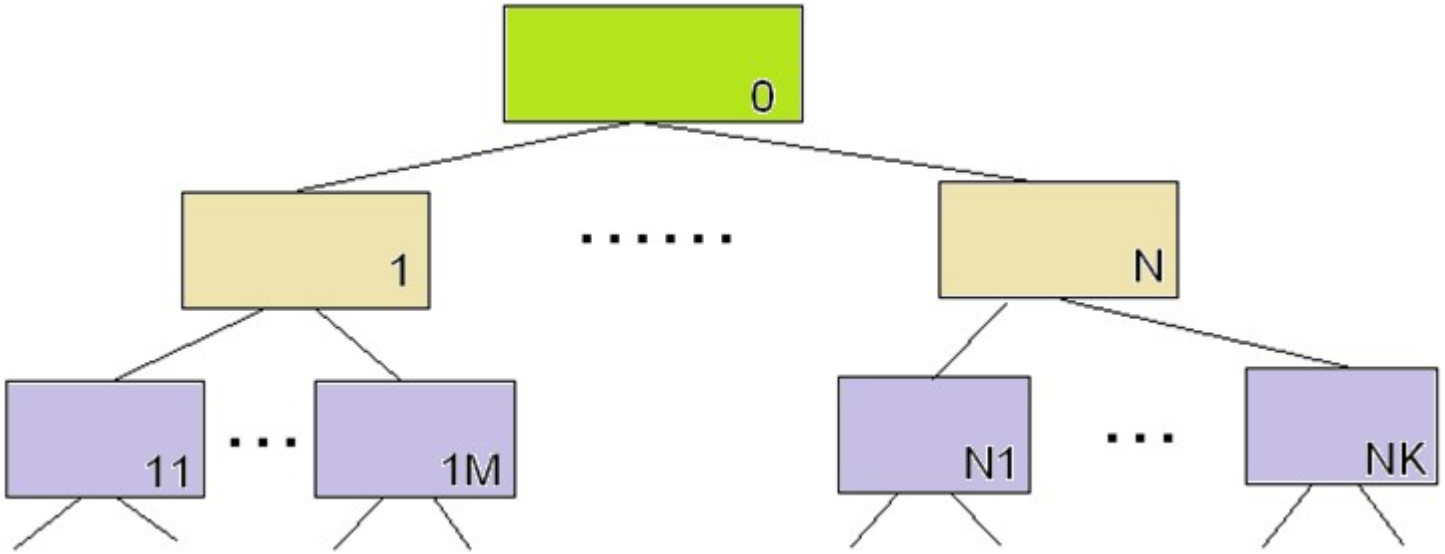
Решение проблемы: ограничение сроков прохождения каждой стадии, основываясь на экспертных оценках.

На каждой итерации оцениваются:

- ✓ риск превышения сроков и стоимости проекта;
- ✓ необходимость выполнения ещё одной итерации;
- ✓ степень полноты и точности понимания требований к системе;
- ✓ целесообразность прекращения проекта.

Нисходящая и восходящая разработка

При проектировании и реализации компонентов, полученных при декомпозиции, могут применяться два подхода: *восходящий* и *нисходящий*.



Восходящий подход

Проектирование и реализация осуществляется снизу-вверх.
(от простого к сложному)

Сначала реализуют модули и функции нижних уровней, затем следующих и т.д.

После отладки компонент нижнего уровня осуществляют их сборку, помещают в библиотеки.

Недостатки:

- ✓ Вследствие неполноты спецификаций несогласованность компонент выявляется на самых последних стадиях, а это может привести к перепроектированию системы;

- ✓ Необходимость разработки тестирующих программ;

- ✓ Из-за позднего проектирования интерфейса невозможно продемонстрировать заказчику для уточнения спецификаций и т.д.

Нисходящий подход

Проектирование и последующая реализация компонентов выполняется «сверху-вниз».

- Вначале проектируют компоненты верхних уровней иерархии, затем следующих и т. д. до самых нижних уровней.
- В той же последовательности выполняют и реализацию компонент.
- При этом компоненты нижних, еще не реализованных уровней заменяют «заглушками», что позволяет тестировать и отлаживать уже реализованную часть.

Методы в рамках нисходящего подхода

Это методы определения последовательности проектирования и реализации компонентов.

Выделяют следующие методы:

- *Иерархический*
- *Операционный*
- *Кобинированный*

***Иерархический* метод** предполагает выполнение разработки строго по уровням, если нет зависимости по данным.

Основная проблема – большое количество сложных заглушек.

***Операционный* метод** связывает последовательность разработки модулей с порядком их использования.

Порядок может быть нарушен, если есть зависимость от данных, т.е. модули ввода данных и вывода результатов должны разрабатываться одними из первых.

Что дает такое нарушение?

Ответ:

Такое нарушение позволяет не проектировать сложную заглушку.

***Комбинированный* метод.**

В создаваемом проекте выделяются ключевые модули, которые необходимо реализовать первыми.

Факторы, влияющие на план последовательности реализации модулей:

- достижимость модуля;
- зависимость по данным;
- возможности выдачи результатов;
- наличие вспомогательных модулей;
- наличие необходимых ресурсов.

При прочих равных:

- сложные модули должны разрабатываться ???? простых.
- если некоторый компонент нижнего уровня используется многими компонентами более высоких уровней, то ….
- компоненты обработки правильных и неправильных данных ???

Нисходящий подход обеспечивает:

- максимально полное определение спецификаций компонентов и их согласованность между собой;
- раннее определение интерфейса пользователя, (заказчик может уточнить требования);
- возможность комплексной отладки на ранних стадиях.

Нисходящий подход обычно используют при объектно-ориентированном программировании.

Подход - *«расширение ядра»*

Предполагает:

- первую очередь проектировать и реализовывать некоторую основу – ядро;
- затем ядро наращивают, комбинируя восходящий и нисходящий методы;
- при создании самого ядра используют один из вышеуказанных подходов.