**Каскадная модель жизненного цикла разработки ПО**

Классическая каскадная модель, несмотря на полученную в последнее время нега­тивную оценку, исправно служила специалистам мно­гие годы. Понимание ее сильных сторон и недостатков улучшает оценочный анализ других, зачастую более эффективных моделей жизненного цикла, основанных на данной модели.

В первые годы практики программирования сначала записывался программный код, а затем происходила его отладка. Общепринятым считалось правило начинать работу не с разработки плана, а с общего ознакомления с продуктом. Без лишних формальностей можно было спроектировать, закодировать, отладить и протестиро­вать ПО еще до того, как оно будет готово к выпуску. Это напоминало процесс, изо­браженный на **рис. 1.**

**Модель «кодирование–устранение ошибок». Она описывается следующим образом: 1) поставить задачу; 2) выполнять ее до успешного завершения либо отмены; 3) проверить результат; 4) повторить при необходимости с 1-го шага.**

В структуре такого процесса есть несколько "неправиль­ностей" (или недостатков). Во-первых, поскольку изначально не существовало офи­циального проекта или анализа, невозможно было узнать о моменте завершения про­цесса. Также отсутствовал способ определения соответствия требованиям относи­тельно достижения качества.

Делать, пока не будет сделано

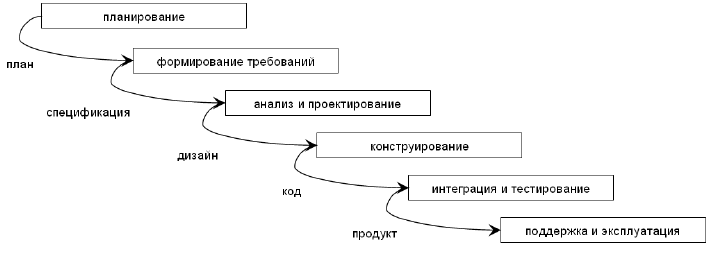
Входы

Выходы

Кодирование и тестирование

***Рис. 1. Модель процесса "делать, пока, не будет сделано”***

В 1970 году каскадная модель была впервые определена как альтернативный вариант метода разработки ПО по принципу *кодирование-устранение ошибок,* который был широко распространен в то время. Это была первая модель, которая формализовала структуру этапов разработки ПО, придавая особое значение исходным требованиям и проектиро­ванию, а также созданию документации на ранних этапах процесса разработки.



***Рис. 2. Классическая каскадная модель***

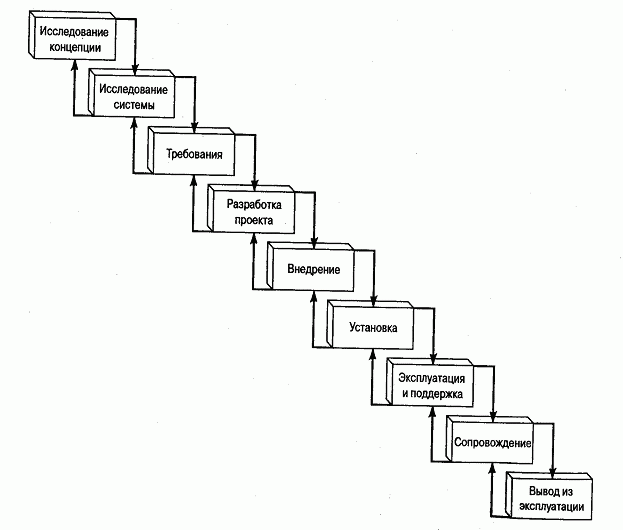
Начальный этап выполнения каскадной модели показан в левой верхней части рис. 2. Продолжение процесса выполнения реализуется с помощью упорядоченной последовательности шагов. В модели предусмотрено, что **каждая последующая фаза начинается лишь тогда, когда полностью завершено выполнение предыдущей фазы** Каждая фаза имеет определенные критерии входа и выхода: входные и выходные данные.

В результате выполнения генерируются внутренние или внешние данные проекта, включай документацию и ПО. Документы по анализу требований впоследст­вии передаются системным специалистам, которые в свою очередь передают их раз­работчикам программных систем более высокого уровня. Программисты передают детальные технические характеристики программистам, которые уже представляют готовый код тестерам.

**Переход от одной фазы к другой осуществляется посредством формального обзора**. Таким образом, клиент получает общее представление о процессе разработки, кроме того происходит проверка качества программного продукта. Как правило, прохождение стадии обзора указывает на договоренность между командой разработчиков и клиентом о том, что текущая фаза завершена и можно перейти к выполнению следующей фазы. Окончание фазы удобно принимать за стадию в процессе выполнения проекта.

Отличительным свойством каскадной модели можно назвать то, что она представ­ляет собой формальный метод, разновидность разработки "сверху вниз", она состоит из независимых фаз, выполняемых последовательно, и подвержена частому обзору.

Спустя непродолжительное время после своего появления на свет каскадная модель была доработана Уинстом Ройсом с учетом взаимозависимости этапов и необходимости возврата на предыдущие ступени, что может быть вызвано, например, неполнотой требований или ошибками в формировании задания. В таком «обратимом» виде каскадная модель просуществовала долгое время и явилась основой для многих проектов (рис. 1).



***Рис. 3. Модифицированная каскадная модель с обратной связью***

**Краткое описание фаз модифицированной каскадной модели**

Приведенная ниже характеристика представляет собой краткое описание каждой фазы каскадной модели (включая фазы интеграции):

* **исследование концепции** — происходит исследование требований на системном уровне с целью определения возможности реализации концепции;
* **процесс системного распределения** — может быть пропущен для систем по раз­работке исключительно ПО. Для систем, в которых необходима разработка как аппаратного, так и программного обеспечения, требуемые функции применяются к ПО и оборудованию в соответствии с общей архитектурой системы;
* **процесс определения требований** — определяются программные требования для информационной предметной области системы, предназначение, линии поведения, производительность и интерфейсы. (В случае необходимости в процесс также включено функциональное распределение системных требований к аппа­ратному и программному обеспечению.);
* **процесс разработки проекта**— разрабатывается и формулируется логически по­следовательная техническая характеристика программной системы, включая структуры данных, архитектуру ПО, интерфейсные представления и процессуаль­ную (алгоритмическую) детализацию;
* **процесс реализации** — в результате его выполнения эскизное описание ПО пре­вращается в полноценный программный продукт. При этом создается исходный код, база данных и документация, которые лежат в основе физического преоб­разования проекта. Если программный продукт представляет собой приобре­тенный пакет прикладных программ, основными действиями по его реализации будут являться установка и тестирование пакета программ. Если программный продукт разрабатывается на заказ, основными действиями являются програм­мирование и код-тестирование;
* **процесс установки** — включает установку ПО, его проверку и официальную приемку заказчиком для операционной среды;
* **процесс эксплуатации и поддержки** - подразумевает запуск пользователем системы и текущее обеспечение, включая предоставление технической помощи, обсуждение возникших вопросов с пользователем, регистрацию запросов пользователя на модернизацию и внесение изменений, а также корректирование или устранение ошибок;
* **процесс сопровождения—** связан с разрешением программных ошибок, неис­правностей, сбоев, модернизацией и внесением изменений, генерируемых про­цессом поддержки. Состоит из итераций разработки и предполагает обратную связь по предоставлению информации об аномалиях;
* **процесс вывода из эксплуатации** — вывод существующей системы из ее активного использования либо путем прекращения ее работы, либо благодаря ее замене но­вой системой или модернизированной версией существующей системы;
* **интегральные задачи** — включают начало работы над проектом, мониторинг про­екта и его управление, управление качеством, верификацию и аттестацию, ме­неджмент конфигурации, разработку документации и профессиональную подго­товку на протяжении всего жизненного цикла.

**Преимущества каскадной модели**

Нетрудно заметить, что каскадная модель имеет множество преимуществ, если ее использовать в проекте, для которого она достаточно приемлема. Ниже перечислены эти преимущества:

* модель хорошо известна потребителям, не имеющим отношения к разработке и экс­плуатации программ, и конечным пользователям (она часто используется другими организациями для отслеживания проектов, не связанных с разработкой ПО);
* она весьма доступна для понимания, так как преследуется простая цель — выпол­нить необходимые действия;
* она проста и удобна в применении, так как процесс разработки выполняется поэтапно;
* ее структурой может руководствоваться даже слабо подготовленный в техниче­ском плане или неопытный персонал;
* она отличается стабильностью требований;
* она представляет собой шаблон, в который можно поместить методы для выпол­нения анализа, проектирования, кодирования, тестирования и обеспечения;
* **она хорошо срабатывает тогда, когда требования к качеству доминируют над тре­бованиями к затратам и графику выполнения проекта;**
* она способствует осуществлению строгого контроля менеджмента проекта;
* при правильном использовании модели дефекты можно обнаружить на более ран­них этапах, когда их устранение еще не требует относительно больших затрат;
* она облегчает работу менеджеру проекта по составлению плана
* она позволяет участникам проекта, завершившим действия на выполняемой ими фазе, принять участие в реализации других проектов;
* она определяет процедуры по контролю за качеством. Каждые полученные дан­ные подвергаются обзору. Такая процедура используется командой разработчиков для определения качества системы;
* стадии модели довольно хорошо определены и понятны;
* ход выполнения проекта легко проследить с помощью использования временной шкалы (или диаграммы Гантта), поскольку момент завершения каждой фазы ис­пользуется в качестве стадии.

**Недостатки каскадной модели**

Но при использовании каскадной модели для проекта, который трудно назвать подходящим для нее, проявляются следующими недостатки:

* в основе модели лежит последовательная линейная структура, в результате чего каждая попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо проблему или недостаток, приведет к значительному увеличению затрат и сбою в графике;
* она может создать ошибочное впечатление о работе над проектом. Выражение типа "35 процентов выполнено" — не несет никакого смысла и не является показа­телем для менеджера проекта;
* интеграция всех полученных результатов происходит внезапно в завершающей стадии работы модели. В результате такого единичного прохода через весь про­цесс, связанные с интегрированием проблемы, как правило, дают о себе знать слишком поздно. Следовательно, проявятся не обнаруженные ранее ошибки или конструктивные недостатки, повысить степень риска при небольшом задаче вре­мени на восстановление продукта;
* у клиента едва ли есть возможность ознакомиться с системой заранее, это проис­ходит лишь в самом конце жизненного цикла. Поскольку готовый продукт не дос­тупен вплоть до окончания процесса, пользователь принимает участие в процессе разработки только в самом начале — при сборе требований, и в конце — во время приемочных испытаний;
* пользователи не могут убедиться в качестве разработанного продукта до оконча­ния всего процесса разработки. Они не имеют возможности оценить качество, ес­ли нельзя увидеть готовый продукт разработки;
* у пользователя нет возможности постепенно привыкнуть к системе. Процесс обуче­ния происходит в конце жизненного цикла, когда ПО уже запущено в эксплуатацию;
* для каждой фазы создаются результативные данные, которые по его завершению считаются замороженными. Это означает, что они не должны изменяться на сле­дующих этапах жизненного цикла продукта. Если элемент результативных данных какого-либо этапа изменяется (что встречается весьма часто), на проект окажет негативное влияние изменение графика, поскольку ни модель, ни план не были рассчитаны на внесение и разрешение изменения на более поздних этапах жиз­ненного цикла;
* все требования должны быть известны в начале жизненного цикла, но клиенты редко могут сформулировать все четко заданные требования на этот момент раз­работки. Модель не рассчитана на динамические изменения в требованиях на про­тяжении всего жизненного цикла, так как получаемые данные "замораживаются". Использование модели может повлечь за собой значительные затраты, если тре­бования в недостаточной мере известны или подвержены динамическим измене­ниям во время протекания жизненного цикла;
* возникает необходимость в жестком управлении и контроле, поскольку в модели не предусмотрена возможность модификации требований;
* модель основана на документации, а значит, количество документов может быть избыточным;
* весь программный продукт разрабатывается за один раз. Нет возможности раз­бить систему на части. В результате взятых разработчиками обязательств разрабо­тать целую систему за один раз могут возникнуть проблемы с финансированием проекта. Происходит распределение больших денежных средств, а сама модель едва ли позволяет повторно распределить средства, не разрушив при этом проект в процессе его выполнения;

**Область применения каскадной модели**

Из-за недостатков каскадной модели ее применение необходимо ограничить ситуа­циями, в которых требования и их реализация максимально четко определены и понятны.

Каскадная модель хорошо функционирует при ее применении в циклах разработ­ки программного продукта, в которых используется неизменяемое определение про­дукта и вполне понятные технические методики.

Если компания имеет опыт построения определенного рода системы — автомати­зированного бухгалтерского учета, начисления зарплаты, ревизии, компиляции, про­изводства, — тогда в проекте, ориентированном на построение еще одного продукта такого же типа, возможно, даже основанного на существующих разработках, можно эффективно использовать каскадную модель. Другим примером надлежащего приме­нения модели может служить создание и выпуск новой версии уже существующего продукта, если вносимые изменения вполне определены и управляемы. Перенос уже существующего продукта на новую платформу часто приводят в качестве идеального примера использования каскадной модели в проекте.

При всей справедливости критики этой модели все же следует признать, что моди­фицированная версия каскадной модели является в значительной степени менее жест­кой, чем ее первоначальная форма. Здесь включаются итерации между фазами, парал­лельные фазы и менеджмент изменений. Обратные стрелки предполагают возмож­ность существования итераций между действиями в рамках фаз. Чтобы отобразить согласованность между этапами, их объединяют прямоугольниками или под прямо­угольниками перечисляют выполняемые на данных этапах действия, чтобы продемон­стрировать согласованность между ними. Несмотря на то, что модифицированная кас­кадная модель является значительно более гибкой, чем классическая модель, она все же не является наилучшим выбором для выполнения проектов по ускоренной разработке.