Каскадная модель

**Каскадная модель** (англ. *waterfall model*) — модель процесса разработки программного обеспечения, жизненный цикл программного обеспечения которой выглядит как поток, последовательно проходящий фазы анализа требований, проектирования. реализации, тестирования, интеграции и поддержки.

Процесс разработки реализуется с помощью упорядоченной последовательности независимых шагов. Модель предусматривает, что каждый последующий шаг начинается после полного завершения выполнения предыдущего шага. На всех шагах модели выполняются вспомогательные и организационные процессы и работы, включающие управление проектом, оценку и управление качеством, верификацию и аттестацию, менеджмент конфигурации, разработку документации. В результате завершения шагов формируются промежуточные продукты, которые не могут изменяться на последующих шагах.

Жизненный цикл традиционно разделяют на следующие основные *этапы*:

1. Анализ требований,
2. Проектирование,
3. Кодирование (программирование),
4. Тестирование и отладка,
5. Эксплуатация и сопровождение.

Схема каскадной модели:

вставленное-изображение.tiff

Достоинства каскадной модели:

* стабильность требований  в течение всего жизненного цикла разработки;
* на каждой стадии формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
* определенность и понятность шагов модели и простота её применения;
* выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие ресурсы (денежные. материальные и людские).

Недостатки модели:

* сложность чёткого формулирования требований и невозможность их динамического изменения на протяжении пока идет полный жизненный цикл;
* низкая гибкость в управлении проектом;
* последовательность линейной структуры процесса разработки, в результате возврат к предыдущим шагам для решения возникающих проблем приводит к увеличению затрат и нарушению графика работ;
* непригодность промежуточного продукта для использования;
* невозможность гибкого моделирования уникальных систем;
* позднее обнаружение проблем, связанных со сборкой, в связи с одновременной интеграцией всех результатов в конце разработки;
* недостаточное участие пользователя в создании системы — в самом начале (при разработке требований) и в конце (во время приёмочных испытаний);
* пользователи не могут убедиться в качестве разрабатываемого продукта до окончания всего процесса разработки. Они не имеют возможности оценить качество, т.к.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;
* каждая фаза является предпосылкой для выполнения последующих действий, что превращает такой метод в рискованный выбор для систем, не имеющих аналогов, т.к. он не поддается гибкому моделированию.

Область применения Каскадной модели

Ограничение области применения каскадной модели определяется её недостатками. Её использование наиболее эффективно в следующих случаях:

1. при разработке проектов с четкими, неизменяемыми в течение *жизненного цикла* требованиями, понятными реализацией и техническими методиками;
2. при разработке проекта, ориентированного на построение системы или продукта такого же типа, как уже разрабатывались разработчиками ранее;
3. при разработке проекта, связанного с созданием и выпуском новой версии уже существующего продукта или системы;
4. при разработке проекта, связанного с переносом уже существующего продукта или системы на новую платформу;
5. при выполнении больших проектов, в которых задействовано несколько больших команд разработчиков.

Инкрементная модель

**Инкрементная модель** (англ. *increment* — увеличение, приращение) подразумевает разработку программного обеспечения с линейной последовательностью стадий, но в несколько инкрементов (версий), т.е. с запланированным улучшением продукта за все время пока Жизненный цикл разработки ПО не подойдет к окончанию.

В начале работы над проектом определяются все основные требования к системе, подразделяются на более и менее важные. После чего выполняется разработка системы по принципу приращений, так, чтобы разработчик мог использовать данные, полученные в ходе разработки ПО. Каждый инкремент должен добавлять системе определенную функциональность.

Схема инкрементной модели:



Достоинства:

* затраты, которые получаются в связи с изменением требований пользователей, уменьшаются, повторный анализ и совокупность документации значительно сокращаются по сравнению с каскадной моделью;
* легче получить отзывы от клиента о проделанной работе — клиенты могут озвучить свои комментарии в отношении готовых частей и могут видеть, что уже сделано. Т.к. первые части системы являются прототипом системы в целом.
* у клиента есть возможность быстро получить и освоить программное обеспечение — клиенты могут получить реальные преимущества от системы раньше, чем это было бы возможно с каскадной моделью.

Недостатки модели:

* менеджеры должны постоянно измерять прогресс процесса. в случае быстрой разработки не стоит создавать документы для каждого минимального изменения версии;
* структура системы имеет тенденцию к ухудшению при добавлении новых компонентов — постоянные изменения нарушают структуру системы. Чтобы избежать этого требуется дополнительное время и деньги на рефакторинг. Плохая структура делает программное обеспечение сложным и дорогостоящим для последующих изменений. А прерванный Жизненный цикл ПО приводит еще к большим потерям.

Спиральная модель

**Спиральная модель:** на каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Особое внимание уделяется начальным этапам разработки — анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов.

Схема спиральной модели:вставленное-изображение.tiff

Данная модель представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипировнаие с целью сочетания преимуществ восходящей и нисходящей концепции, делающая упор на начальные этапы жизненного цикла: анализ и проектирование. *Отличительной особенностью* этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла.

На этапах анализа и проектирования реализуемость технических решений и степень удовлетворения потребностей заказчика проверяется путем создания прототипов. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы.

Достоинства модели:

* позволяет быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований;
* допускает изменение требований при разработке программного обеспечения, что характерно для большинства разработок, в том числе и типовых;
* в модели предусмотрена возможность гибкого проектирования, поскольку в ней воплощены преимущества каскадной модели, и в то же время разрешены итерации по всем фазам этой же модели;
* позволяет получить более надежную и устойчивую систему. По мере развития программного обеспечения ошибки и слабые места обнаруживаются и исправляются на каждой итерации;
* эта модель разрешает пользователям активно принимать участие при планировании, анализе рисков, разработке, а также при выполнении оценочных действий;
* уменьшаются риски заказчика. Заказчик может с минимальными для себя финансовыми потерями завершить развитие неперспективного проекта;
* обратная связь по направлению от пользователей к разработчикам выполняется с высокой частотой и на ранних этапах модели, что обеспечивает создание нужного продукта высокого качества.

Недостатки модели:

* если проект имеет низкую степень риска или небольшие размеры, модель может оказаться дорогостоящей. Оценка рисков после прохождения каждой спирали связана с большими затратами;
* Жизненный цикл модели имеет усложненную структуру, поэтому может быть затруднено её применение разработчиками, менеджерами и заказчиками;
* спираль может продолжаться до бесконечности, поскольку каждая ответная реакция заказчика на созданную версию может порождать новый цикл, что отдаляет окончание работы над проектом;
* большое количество промежуточных циклов может привести к необходимости в обработке дополнительной документации;
* использование модели может оказаться дорогостоящим и даже недопустимым по средствам, т.к. время. затраченное на планирование, повторное определение целей, выполнение анализа рисков и прототипирование, может быть чрезмерным;
* могут возникнуть затруднения при определении целей и стадий, указывающих на готовность продолжать процесс разработки на следующей итерации

Область применения спиральной модели

* при разработке проектов, использующих новые технологии;
* при разработке новой серии продуктов или систем;
* при разработке проектов с ожидаемыми существенными изменениями или дополнениями требований;
* для выполнения долгосрочных проектов;
* при разработке проектов, требующих демонстрации качества и версий системы или продукта через короткий период времени;
* при разработке проектов. для которых необходим подсчет затрат, связанных с  оценкой и разрешением рисков.