Согласно этой модели, проект реализуется пошагово, в соответствии с точной последовательностью действий: сбор и изучение требований, проектирование программного обеспечения и его разработка, тестирование и техническая поддержка. Каскадная модель достаточно гибкая, и некоторые этапы могут частично совпадать.

**1. Анализ требований**

На этом этапе важно задокументировать все требования к будущему программному обеспечению. Необходимо посвятить достаточно времени обсуждению деталей проекта со всеми заинтересованными сторонами. Все поступающие данные нужно проанализировать и систематизировать. Важно также учесть все технические ограничения, которые могут возникнуть на стороне заказчика. Итогом данного этапа должно стать создание подробной спецификации, отвечающей всем требованиям заказчика. Также следует обратить внимание и на другие факторы, которые могут затруднять процесс разработки. К ним относятся дедлайны, установленные заказчиком, а также бюджетные ограничения.

*Обратите внимание: Чем больше информации о проекте вы соберете, тем меньше времени потратите на исправление ошибок, доработку проекта, пересмотры бюджета, обсуждения и решение других вопросов.*

**Видение проекта**

Важной задачей является создание подробного *документа видения (или образа) проекта*, который включает краткое описание проекта, бизнес-цели, а также критерии успеха проекта, факторы бизнес-рисков и описание конечного пользователя продукта.

Готовый документ необходимо передать на утверждение заказчику, чтобы убедиться в том, что все поставленные требования были учтены, а также чтобы проинформировать его о любых рисках, которые могут возникнуть после релиза проекта.

**Сбор требований**

После того, как все основные вопросы решены, рекомендуется провести дополнительные обсуждения и интерактивные семинары со всеми заинтересованными сторонами. Это поможет выявить какие-либо неочевидные моменты, которые в дальнейшем могут стать причиной внесения изменений в интерфейс приложения или необходимости переписывания паттернов кода. Данный этап может также включать заполнение анкет, рассмотрение кейсов, мозговой штурм и т.д.

Многие проекты заходят в тупик из-за дополнительных требований, которые всплывают на стадии разработки. Поэтому очень важно понимать начальные бизнес-цели и главную идею будущего приложения.

**2. Проектирование программного обеспечения**

Следующим этапом жизненного цикла ПО является создание документа, описывающего масштабы и границы проекта. Данный документ включает в себя мокапы или скетчи интерфейса будущего приложения, а также подробную спецификацию требований программного обеспечения. Необходимо отметить, что в некоторых случаях документ видения (образа) проекта и документ о масштабах и границах проекта могут быть представлены как единый документ “Об образе и границах проекта”.

**Масштабы и границы проекта**

В документе, описывающем масштабы и границы проекта, должны быть перечислены основные функции создаваемого программного обеспечения. Они определяются на основании документа видения проекта, и безусловно, с учетом указанных временных рамок и установленного бюджета.  
Кроме того, в данный документ входят мокапы или скетчи, созданные на основе документа видения проекта, а также собранных требований.  
Вы можете нарисовать скетч пользовательского интерфейса от руки либо использовать для этого программы создания мокапов, и затем согласовать его с заказчиком. Ниже представлен список полезных программ для создания мокапов, которые мы используем на практике:

<https://invisionapp.com/>  
<https://webflow.com/>  
<https://moqups.com/>

В процессе обсуждения проекта, у заказчика может появляться все больше новых идей относительно его реализации. Поэтому рекомендуется дать ему время на обдумывание своего проекта и требований к нему, а затем повторно собраться и обсудить детали проекта, чтобы ничего не упустить из вида. На этом этапе также поднимается вопрос о послепродажном обслуживании продукта. Вы должны уведомить заказчика о том, каким образом будет осуществляться техническая поддержка после завершения этапа тестирования и последующего релиза продукта. Обратите внимание на то, что документ видения проекта и документ о масштабах и границах проекта должны быть созданы до подписания контракта.

**Спецификация требований программного обеспечения**

Спецификация требований программного обеспечения (SRS) описывает требования, которым должно отвечать создаваемое программное обеспечение. Она должна быть логичной, последовательной, доступной и полной. Требования могут выражаться в разных формах, например, в виде традиционных утверждений долженствования (н-р, “Система Staff Manager должна поддерживать следующие браузеры: Google Chrome, Apple Safari, Mozilla Firefox, Opera, IE 8+”) или в виде пользовательских историй (н-р, “поскольку я являюсь менеджером, мне необходим доступ к персональной информации всех сотрудников”). Существует большое количество шаблонов спецификаций. Выбор определенного шаблона зависит от специфики проекта. В большинстве случаев, спецификация включает в себя описание продукта, классы пользователей, функциональные и нефункциональные требования к разрабатываемому программному обеспечению. Иногда в шаблон также входит прототип. Главное — сделать спецификацию понятной, лаконичной и полезной для разработчиков.

Для создания прототипа вам необходимо выяснить следующее:

* способ получения и обработки входящих данных для создания необходимых данных на выходе;
* форма, в которой должны быть представлены выходные данные.

Мокапы (или прототипы) передаются UI/UX-дизайнерам, которые превращают их в красочные шаблоны.

**3. Разработка программного обеспечения**

Необходимо отметить, что [разработка программного обеспечения](https://xbsoftware.ru/uslugi/#web-razrabotka) может также включать в себя создание интерактивного прототипа, который, в сущности, является основой будущего приложения. Такой прототип помогает определить архитектуру системы в целом. На данном этапе пишется мало кода: например, код кнопок и простых форм, чтобы дать заказчику общее представление о том, как будет работать конечный продукт. Поэтому мы включили создание прототипа в этап разработки программного обеспечения.

Как только интерактивный прототип и дизайн приложения готов и утвержден заказчиком, начинается разработка стандартов приложения (конвенции наименований, способа документирования кода, инструкций для конечного пользователя и т.д.). После этого можно смело переходить к следующему этапу жизненного цикла, а именно, к разработке программного обеспечения. Разработка ПО может быть разделена на небольшие части, или юниты, и каждый юнит разрабатывается и тестируется разработчиками для проверки его функциональности (модульное тестирование).

## 4. Тестирование программного обеспечения

После завершения этапа разработки продукт должен пройти тщателное [тестирование](https://xbsoftware.ru/uslugi/#kontrol-kachestva), чтобы убедиться в том, что он соответствует поставленным требованиям. На этапе приемочного тестирования необходимо, чтобы заказчик попытался применить продукт локально точно таким же образом, как он собирается использовать его после релиза. Когда будут исправлены основные ошибки, программное обеспечение можно внедрять. Для исправления незначительных ошибок может использоваться простая система отслеживания, что позволит исправлять любые недоработки уже на этапе сопровождения ПО.

## 5. Техническая поддержка программного обеспечения

После того, как продукт был протестирован и развернут на сервере заказчика, начинается следующая фаза жизненного цикла разработки программного обеспечения, которая называется сопровождением или технической поддержкой ПО. В целом, сопровождение подразумевает под собой исправление мелких багов, которые обнаруживаются на этом этапе.  
Тем не менее, вполне возможно, что вам придется вносить некоторые изменения в созданное программное обеспечение, несмотря на все усилия, приложенные вами на предыдущих этапах. Заказчик может решить внести изменения в функциональность разработанного продукта. Следовательно, вам придется собирать, описывать и обсуждать новые требования с заказчиком, чтобы внести в продукт необходимые изменения. В данном случае, вам предстоит работа с новым каскадным проектом, и все вышеописанные шаги придется повторять с начала.

**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;