# 1. Жизненный цикл программного обеспечения

## 1.1. Особенности разработки программ и программного обеспечения

Понятия «программа» и «программное обеспечение» (ПО) очень тесно связаны. Четкую границу между ними провести нельзя, однако понимание различий между ними важно, потому что они имеют существенно отличные подходы к их разработке.

Большая программа, состоящая из нескольких взаимодействующих программ, похожа на ПО, однако между ними существует разница в объеме и в степени внутренней связи. Программа имеет относительно меньший объем, чем ПО, и более жесткие связи. В связи с этим, термин ***программное обеспечение*** используют для обозначения большой группы взаимосвязанных и взаимодействующих программ. Термин ***программа*** используют для обозначения относительно меньшего по размерам, более тесно связанного списка команд, которые выполняют более унифицированную функцию.

Развитие вычислительной техники приводит к постоянному возрастанию сложности программного обеспечения. Для разработки малого программного проекта (или небольшой программы) достаточно одного программиста, который обсуждает основные идеи с конечным пользователем. Некоторые идеи он заносит на бумагу и реализует проект в течение нескольких дней или недель. Совершенно иначе выглядят проекты, в которых задействовано много разработчиков, и сроки исполнения исчисляются месяцами и годами. Современные крупные проекты характеризуются следующими особенностями:

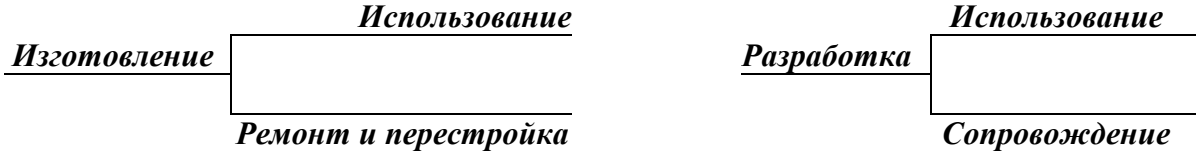
- сложность описания (большое количество функций, данных, процессов и связей между ними), что требует тщательного анализа и моделирования;
- наличие большого количества тесно взаимодействующих компонентов;
- отсутствие прямых аналогов, что не позволяет использовать какие-либо типовые решения;
- необходимость интеграции разработанных и уже существующих приложений;
- функционирование в неоднородной среде и на разных аппаратных платформах;
- разобщенность и разнородность отдельных групп разработчиков;
- существенная временная протяженность проекта;

- постоянное уточнение и изменение требований пользователя.

Разработка малых проектов может выполняться на интуитивном уровне с применением неформальных методов, основанных на искусстве, практическом опыте и дорогостоящих экспериментальных проверках качества программ. В то время как для больших проектов такой подход неприменим. Для этого существуют строгие методологии создания и реализации программ.

## 1.2. Жизненный цикл программного обеспечения. Модели жизненного цикла программного обеспечения

Как и любое изделие, программа имеет свой ***жизненный цикл*** (ЖЦПО), т.е. интервал времени от момента возникновения необходимости в программе до момента прекращения ее использования. В общем виде жизненный цикл любого изделия состоит из трех фаз: изготовление, использование, ремонт и перестройка (рис. 1.1, а). Нужно отметить, что чем больше затрат вложено в изделие, тем больше внимания уделяется фазе «Ремонт и перестройка». Для простого изделия, например, зубочистки данная стадия практически отсутствует, в то время как на ремонт и поддержание большого здания затрачиваются значительные усилия.




а) б)

Рис.1.1. Взаимосвязь фаз жизненного цикла:
а) изделия, б) программы

Программа имеет те же фазы жизненного цикла, но, учитывая специфику создания программ, их называют иначе: разработка, использования, сопровождение (рис.1.1, б). В отличие от других изделий определяющей стадией для программ является сопровождение. Статистика показывает, что на разработку программ затрачивается 50 % времени и 38 % трудозатрат [1], остальные же затраты приходятся на стадию сопровождения. Это происходит по двум причинам. Во-первых, хорошая программа требует больших затрат, поэтому при ее создании сразу планируется возможность дальнейшего изменения. Во-вторых, программы проще модифицировать. Практически программы могли бы существовать вечно, так как их физического износа не происходит, однако из-за морального устаревания они все

же заканчивают свой жизненный цикл. Считают, что программа морально устарела, если она перестала удовлетворять актуальным требованиям, и ее дальнейшая модификация не целесообразна.

Под ***моделью ЖЦПО*** понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦПО. Модель ЖЦПО зависит от специфики ПО и условий, в которых оно создается и функционирует.

Модель, приведенная на рис. 1.1, носит название «Обобщенная модель ЖЦПО», кроме нее известны каскадная (70-85 гг.) и спиральная (86-90 гг.) модели.

## 1.3. Характеристика обобщенной модели ЖЦПО

Рассмотрим фазы обобщенной модели ЖЦПО: использование, разработку и сопровождение.

### 1.3.1. Фаза «Использование»

Несмотря на то, что фаза использования следует за фазой разработки (рис. 1.1), именно она является определяющей, так как ее характеристики влияют на другие фазы ЖЦПО.

Перечислим основные характеристики фазы использования.

1. ***Тип использования ПО***: диалоговый (например, служащий трансагенства) или автономный (например, бухгалтер фирмы). В случае диалогового режима особое внимание приходится обращать на интерфейсную часть.

2. ***Периодичность использования ПО*** (например, крайне важно учесть эффективность использования памяти при разработке программы, работающей постоянно, и не так важно - для программы, работающей раз в год).

3. ***Последствия отказов*** - при разных типах использования, иногда отказ приемлем, иногда – нет.

4. ***Количество и уровень пользователей*** (например, пятьсот пользователей, использующих программное обеспечение по-разному, влияют на его разработку совершенно иначе, чем пятьсот «одинаковых пользователей»).

5. ***Тип программного обеспечения*** (прикладное, системное и т.п.).

6. ***Масштабность*** (например, за один день возможно написать бухгалтерскую задачу, в то время как программное обеспечение по управлению спутником пишется годы сотнями людей).

7. ***Сложность*** бывает техническая и логическая (много логических переходов).

8. ***Ясность*** - существуют приложения, которые трудно запрограммировать, например, качество нефти определяется на вкус.

9. ***Является ПО продуктом или проектом***, проект отличается от продукта двумя аспектами: определением требований и широтой функций. ***Определяя требования*** к продукту нужно не просто удовлетворить требования большого числа пользователей, но и выиграть соревнование с другими системами, при этом тесных связей с пользователем нет. В случае проекта такая связь существует, в этом случае важно удовлетворить требования конкретного заказчика. Проект отличается от продукта также ***широтой функций,*** например, для бухгалтерских задач новую операционную систему писать не нужно, однако для космического проекта это, скорее всего, потребуется.

### 1.3.2. Фаза «Разработка»

Разработка ПО может вестись с использованием каскадной (см. пункт 1.4) или итеративной (см. пункт 1.5) моделей ЖЦПО.

### 1.3.3. Фаза «Сопровождение»

***Сопровождение*** – это процесс модификации ПО с целью исправления выявленных ошибок и изменения его функциональных возможностей.

Существует два типа сопровождения - в узком и в широком смысле.

В узком смысле сопровождение ПО - это поддержка его в актуальном состоянии в условиях изменяющейся среды функционирования, а также обеспечение его работоспособности.

В широком смысле сопровождение ПО включает в себя следующие виды деятельности:

- планирование и координация разработки ПО;
- исправление ошибок;
- модификация функций и добавление новых;
- разработка средств диагностики;
- разработка средств обучения;
- организация рекламы;
- тиражирование ПО и документации к нему;
- поставка ПО пользователям, оказание услуг по внедрению и эксплуатации;
- обучение пользователей работе с ПО.

## 1.4. Характеристика каскадной модели ЖЦПО

Долгое время программное обеспечение пытались разрабатывать, следуя каскадной модели (лавинообразная модель, модель «водопада» (waterfall)). В соответствии с ней необходимо было сначала проанализировать требования к будущей системе, спроектировать, создать и протестировать систему, а затем установить ее у пользователей. Выделяют следующие этапы разработки ПО с использованием данного подхода:

- анализ;
- проектирование;
- реализация;
- отладка и тестирование;
- внедрение;
- сопровождение.

Согласно каскадной модели переход с одного этапа на следующий должен осуществляться только после того, как будет полностью завершена работа на текущем этапе (рис. 1.2). В завершении каждого этапа должен выпускаться полный комплект документации, достаточный для того, чтобы разработка могла быть продолжена другой командой разработчиков.

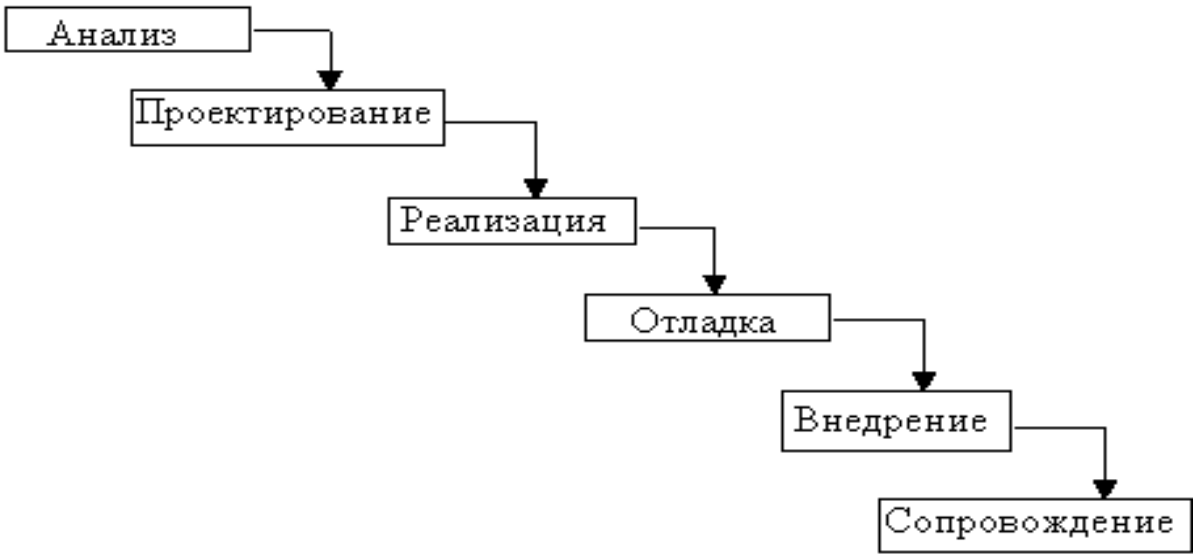



Рис.1.2. Каскадная модель разработки ПО

Рассмотрим содержание основных этапов каскадной модели ЖЦПО.

Целью этапа ***«Анализ»*** является описание задачи, которое должно быть полным, последовательным, доступным для чтения и обзора различными заинтересованными сторонами, а также позволяющим производить сравнение с реальными условиями. В ходе этого этапа уточняются требования, приведенные в задании на проектирование, и разрабатываются спецификации на ПО. Итогом выполнения этого этапа являются эксплуатаци-

онные и функциональные спецификации, которые содержат конкретное описание ПО. ***Эксплуатационные спецификации*** должны содержать сведения о быстродействии ПО, затратах памяти, требуемых технических средствах, надежности и т.п. ***Функциональные спецификации*** определяют функции, которые должно выполнять ПО. Спецификации должны быть полными, точными и ясными.

Целью этапа ***«Проектирование»*** является иерархическое разбиение сложной задачи по созданию ПО на подзадачи меньшей сложности. На этом этапе выполняется следующая деятельность:

- выбор структуры информации;
- формирование структуры ПО и разработка алгоритмов, которые задаются спецификацией;
- определение состава модулей с разделением их на иерархические уровни;
- фиксация межмодульных интерфейсов.

Результатом проектирования является разбиение системы на отдельные модули, дальнейшая декомпозиция которых нецелесообразна.

Следующий этап - ***«Реализация».***

Целью этапа ***«Отладка и тестирование»*** является выявление ошибок, проверка работоспособности ПО и его соответствие спецификации. В ходе этого этапа решаются следующие задачи:

- подготовка данных для отладки;
- планирование отладки;
- разработка тестов;
- испытание ПО.

Результатом данного этапа является протестированное и отлаженное ПО.

***«Внедрение»*** – это процесс передачи ПО заказчику. Этап ***«Сопровождение»*** был рассмотрен в пункте «Характеристика обобщенной модели ЖЦПО».

Каскадный подход хорошо зарекомендовал себя при построении однородных систем, в которых каждое приложение представляет собой единое целое и для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования.

Положительные стороны применения каскадного подхода заключаются в следующем:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логичной последовательности этапы работ позволяют планировать сроки их завершения и соответствующие затраты.

Каскадный подход был официальной методологией, применявшейся во многих проектах. Однако можно с уверенностью сказать, что в чистом

виде он не использовался ни разу. Одним из главных недостатков модели «водопада» является невозможность возврата назад на пройденные этапы.

Рассмотрим трудности, которые возникают, если пытаться строго следовать каскадной модели.

Вначале проекта перед разработчиками стоит практически не осуществимая задача полностью определить все требования к системе. Однако, несмотря на то что процессы, которые должна поддерживать проектируемая система, тщательно и всесторонне обсуждаются с пользователями и исследуются реально на стадии анализа удается собрать не более 80 % требований к системе.

Затем начинается этап проектирования. Необходимо определить архитектуру будущей системы. При этом могут обнаружиться новые проблемы, их необходимо опять обсуждать с пользователями, что выразится в появлении новых требований к системе. Итак, приходится возвращаться к анализу. После нескольких таких уточнений, наконец, наступает этап написания программного кода.

На этом этапе обнаруживается, что некоторые из принятых ранее решений невозможно осуществить. Приходится возвращаться к проектированию и пересматривать эти решения. После завершения кодирования наступает этап тестирования. Здесь выясняется, что требования не были достаточно детализированы и их реализация некорректна. Нужно возвращаться назад на этап анализа и пересматривать эти требования.

Наконец система готова и поставлена пользователям. Поскольку прошло уже много времени и условия, вероятно, уже изменились, пользователи воспринимают ее без большого энтузиазма, отвечая примерно так: «Да, это то, что я просил, но не то, что я хочу!»

Таким образом, можно выделить следующие недостатки каскадного подхода.

1. Реальный процесс создания ПО никогда полностью не укладывался в жесткую схему каскадной модели. В процессе создания ПО постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате, реальный процесс создания ПО принимает вид, представленный на рис. 1.3.

2. При каскадном подходе наблюдается запаздывание с получением результатов. Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ. Требования к ПО «заморожены» в виде технического задания на все время его создания. Поэтому в случае неточного изложения требований или их изменения в течение длительного периода создания ПО, пользователи получают систему, не удовлетворяющую их потребностям. Модели как функциональные, так и информационные могут устареть одновременно с их утверждением.

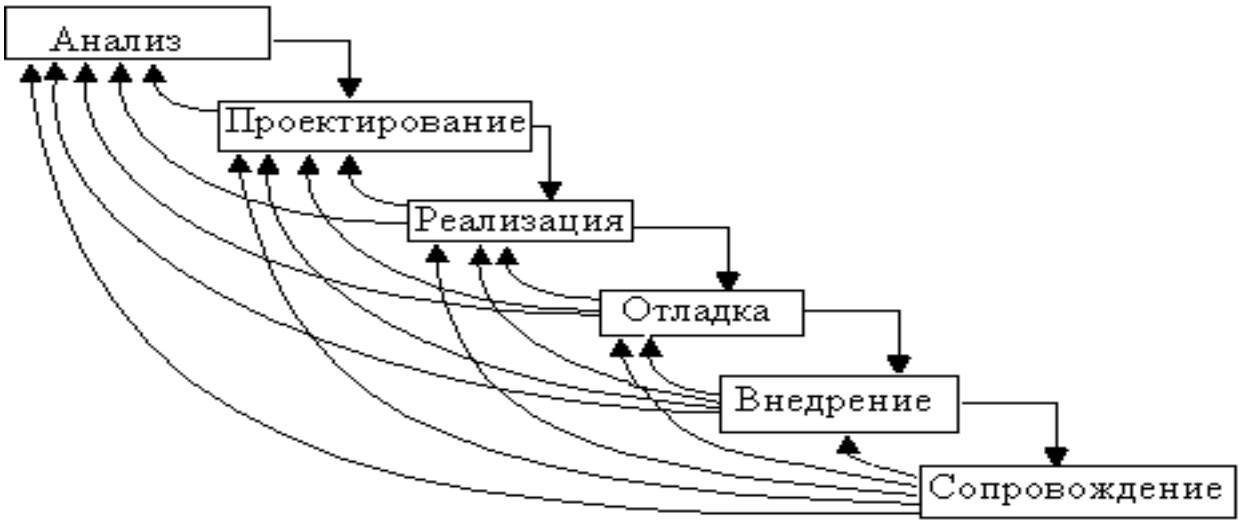



Рис. 1.3. Реальный процесс разработки в соответствии с каскадной моделью

## 1.5. Характеристика спиральной модели ЖЦПО

Любая человеческая деятельность, которая требует творчества и инноваций, идет путем проб и ошибок, итеративно развивающегося процесса. Разработка программного обеспечения – сложный процесс, и его аккуратное поэтапное выполнение не всегда возможно. Если игнорировать необходимость возврата, то система будет содержать дефекты проектирования, некоторые требования будут не учтены, возможны и более серьезные последствия.

Для преодоления проблем, характерных каскадной модели, была выработана спиральная модель ЖЦПО (рис. 1.4), поддерживающая итеративный процесс разработки. В ней возврат на предыдущие этапы планируется заранее. Анализ, проектирование, реализация, отладка и установка системы выполняются по несколько раз. В рамках такой концепции проект можно рассматривать как последовательность небольших «водопадиков».

Спиральная модель обладает следующими характеристиками.

1. Итеративная разработка большое значение придает начальным этапам: анализ и проектирование. Невозможно сразу выявить все требования к системе и создать идеальную архитектуру. Нужно учитывать появление новых требований и изменение архитектуры в процессе разработки, планируя несколько итераций.

Итеративный процесс предполагает, что требования к системе и созданная архитектура вновь и вновь подвергаются анализу и проектированию. При этом в каждом цикле анализ-проектирование-эволюция принятые решения развиваются, приближаясь к требованиям конечного пользо-

вателя (часто даже не высказанным), оставаясь при этом простыми, надежными и открытыми для дальнейшего изменения.

2. Реализуемость технических решений проверяется путем создания прототипов системы. Каждый виток спирали соответствует созданию версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Детали проекта постепенно углубляются и последовательно конкретизируются, в результате чего выбирается обоснованный вариант, который доводится до реализации. Главная же задача – как можно быстрее показать пользователям очередную версию системы, тем самым активизируя процесс уточнения и дополнения требований.

3. Разработка итерациями отражает объективно существующий спиральный цикл создания систем. На каждом этапе не требуется полного завершения работ. Это позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем; недостающая работа выполняется на следующей итерации.

4. Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов жизненного цикла. Переход осуществляется в соответствии с планом. План составляется на основе статистических данных, полученных в предыдущих проектах и личного опыта разработчиков.

Итеративная модель ЖЦПО в большей степени, чем каскадная гарантирует, что созданная система оправдает ожидания заказчика, уложится во временные и финансовые рамки, будет легко поддаваться изменению и адаптации.

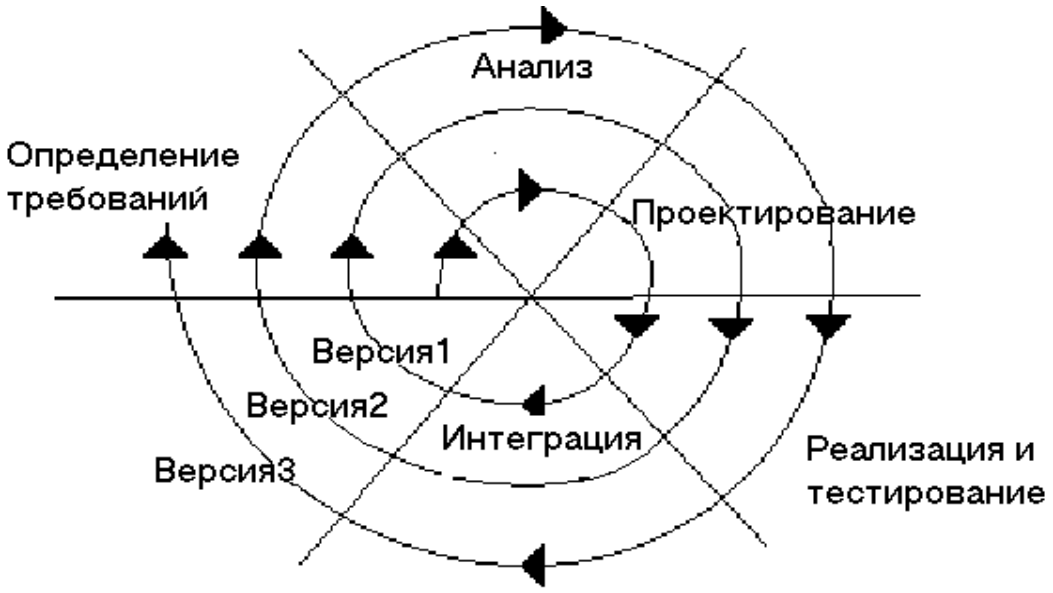



Рис 1.4. Спиральная модель ЖЦПО