Разработка кода информационных систем

Лекция №1-2 «Информационная система. Жизненный цикл и этапы конструирования программ»

Старший преподаватель кафедры автоматизированных информационных систем ОВД
Воронежского института МВД России
подполковник полиции
Петрова Елена Владиленовна

Учебные вопросы:

1. Информационная система: основные понятия.
2. Обзор моделей жизненного цикла программного обеспечения.
3. Процессы жизненного цикла программного обеспечения.
4. Единая система программной документации.

1. Информационная система: основные понятия

Информационная система – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы, которые обеспечивают и распространяют информацию.

ИС предназначена для удовлетворения конкретных информационных потребностей в рамках определенной предметной области.

Информационная продукция – документы, информационные массивы, базы данных и информационные услуги.

1. Информационная система: основные понятия

Компоненты ИС

В широком смысле:

- данные;
- техническое обеспечение;
- программное обеспечение;
- организационное обеспечение;
- персонал.

В узком смысле:

- данные;
- аппаратное обеспечение;
- программное обеспечение.

1. Информационная система: основные понятия

Информационная система – автоматизированная система, результатом функционирования которой является представление выходной информации для последующего использования. (ГОСТ РВ 51987)

Автоматизированная система – система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций. (ГОСТ 34.003-90)

Информационная технология – процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления этих процессов и методов. (ГОСТ Р ИСО/МЭК 12119-2000)

1. Информационная система: основные понятия

Информационно-вычислительная система – совокупность данных (или баз данных), систем управления базами данных и прикладных программ, функционирующих на вычислительных средствах как единое целое для решения определенных задач.
(ГОСТ Р 53622-2009)

2. Обзор моделей жизненного цикла программного обеспечения

Жизненный цикл программного обеспечения – период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Состав процессов жизненного цикла регламентируется стандартом **ГОСТ Р ИСО/МЭК 12207-2010** и соответствующим ему международным стандартом ISO/IEC 12207: 2008.

2. Обзор моделей жизненного цикла программного обеспечения

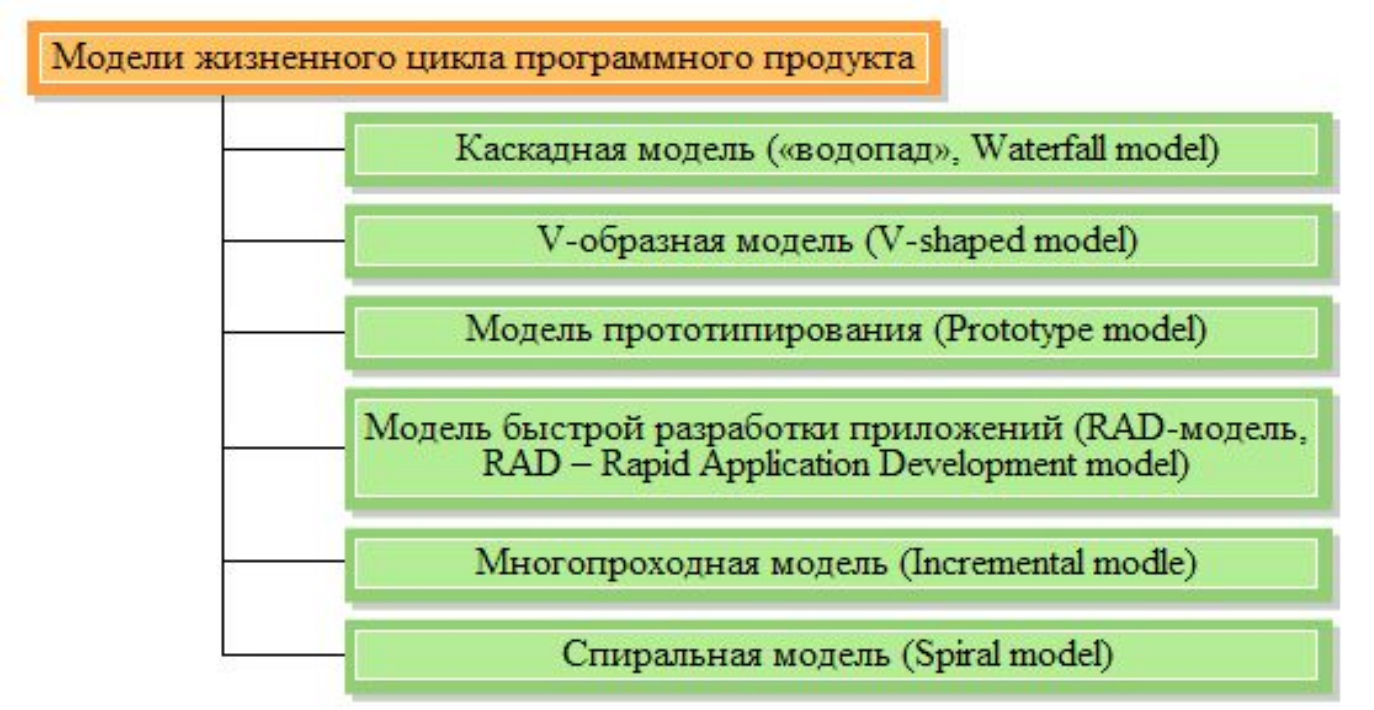
Жизненный цикл программного продукта включает в себя :

- **Разработку;**
- **Развертывание** - если программный продукт достаточно сложный, то его развертывание у клиентов, как правило, реализуется отдельными самостоятельными проектами внедрения;
- **Поддержку** - заключается в разработке новой функциональности, переработке уже существующей функциональности в связи с изменением требований и улучшением продукта, а также устранение некритических замечаний к ПО, выявленных при его эксплуатации.

•**Сопровождение** - включает в себя устранение критических неисправностей в системе и реализуется часто не как проект а, как процессная деятельность.

2. Обзор моделей жизненного цикла программного обеспечения

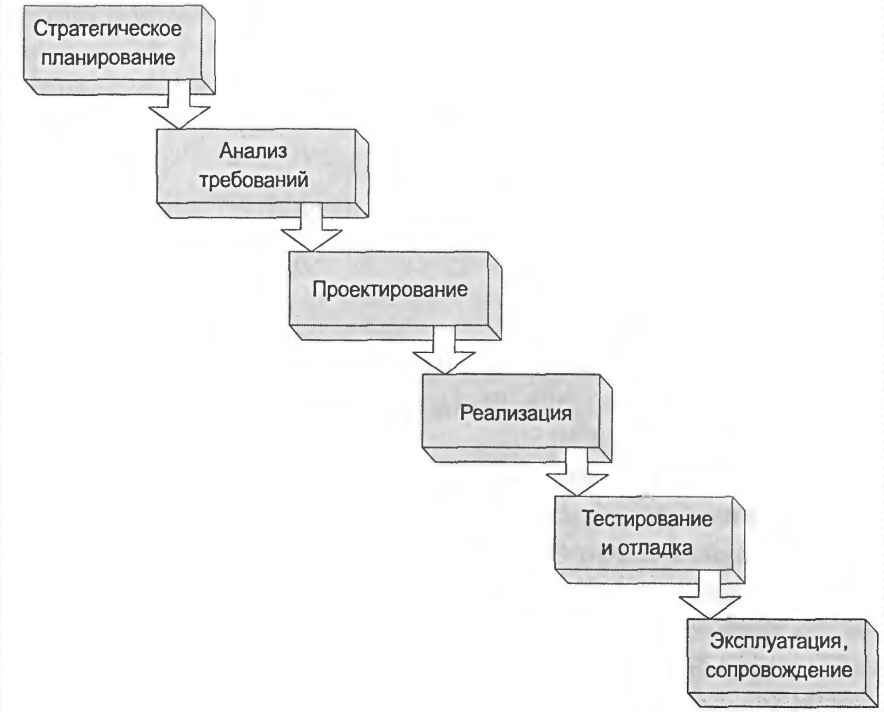
Модель жизненного цикла - структура, состоящая из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающая жизнь системы от установления требований к ней до прекращения ее использования.



Наиболее распространенные модели жизненного цикла

2. Обзор моделей жизненного цикла программного обеспечения

Каскадная модель характеризуется следующими основными особенностями:

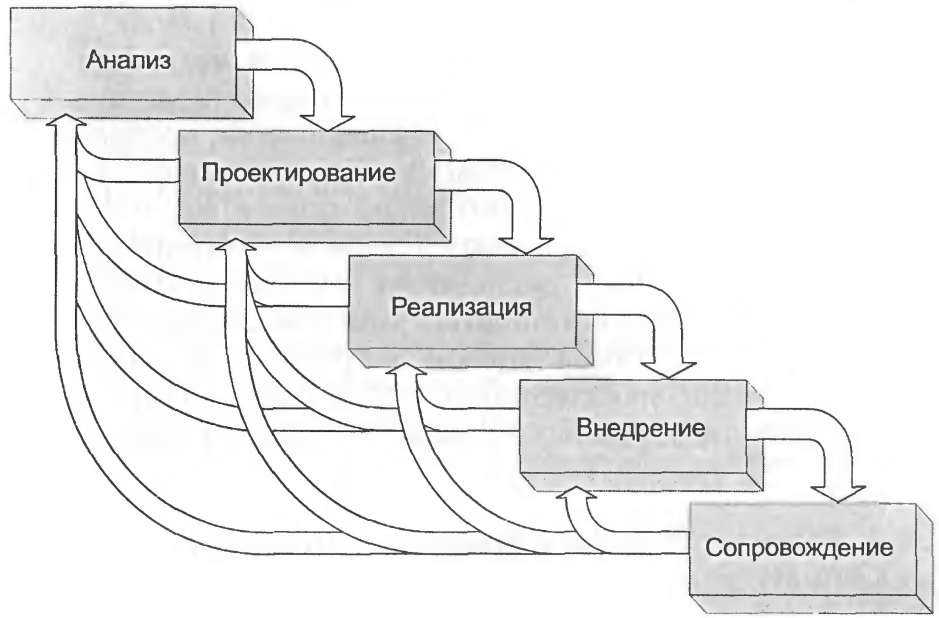


- последовательным выполнением входящих в ее состав этапов;
- окончанием каждого предыдущего этапа до начала последующего;
- отсутствием временного перекрытия этапов;
- отсутствием (или определенным ограничением) возврата к предыдущим этапам;
- наличием результата только в конце разработки.

Выявление и устранение ошибок в каскадной модели производится только на стадии тестирования, которая может растянуться во времени или вообще никогда не завершиться.

2. Обзор моделей жизненного цикла программного обеспечения

Итерационная модель жизненного цикла ПО (водоворот, поэтапная модель с промежуточным контролем) характеризуется наличием обратных связей между этапами, вследствие этого появляется возможность проведения проверок и корректировок проектируемой ИС на каждой стадии разработки.

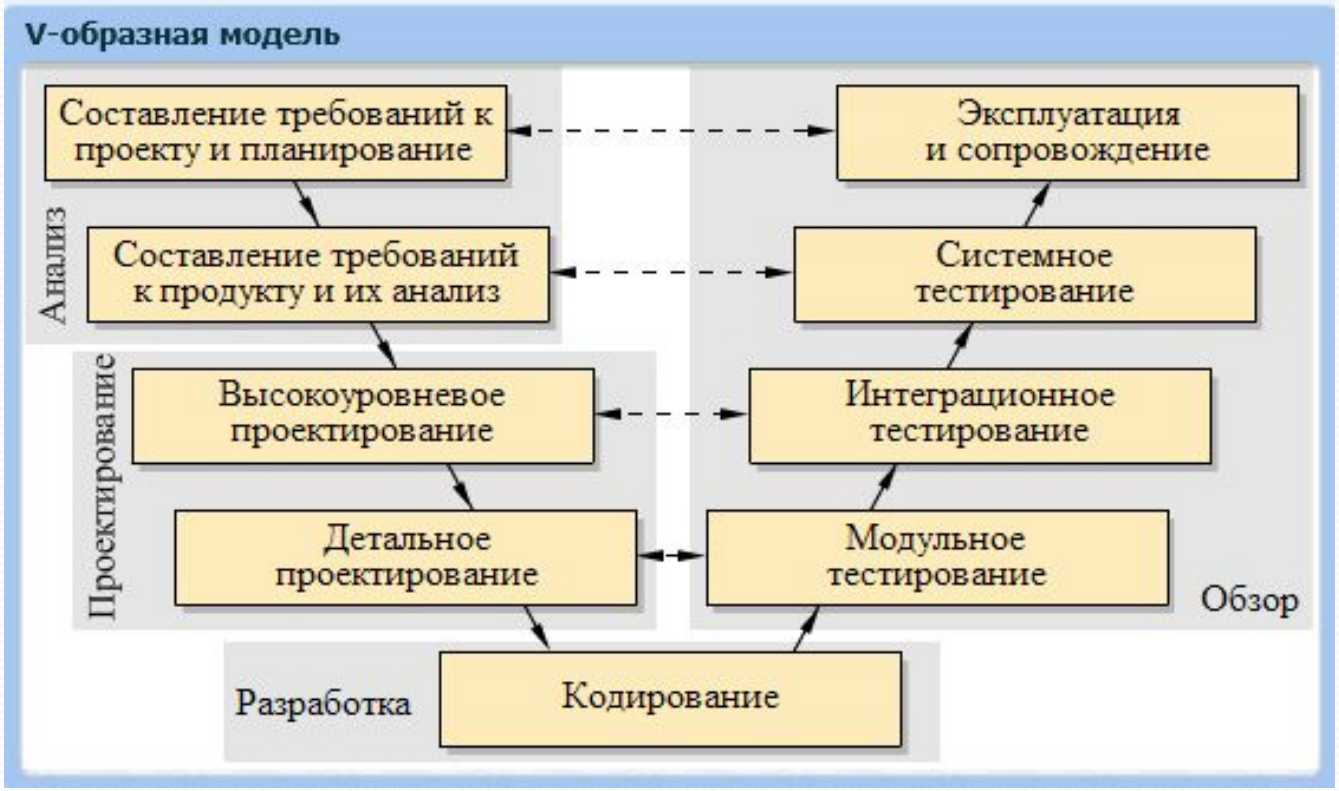


В результате трудоемкость отладки по сравнению с каскадной моделью существенно снижается.

Основной недостаток: в процессе разработки системы могут измениться начальные требования, и в этом случае итерационная модель может оказаться неэффективной.

2. Обзор моделей жизненного цикла программного обеспечения

В **V-образной модели** тестирование обсуждается, проектируется и планируется уже на ранних этапах разработки. Каждая последующая фаза начинается только после успешного завершения предыдущей. Для решения проблемы определяется четыре базовых шага:



2. Обзор моделей жизненного цикла программного обеспечения

В **V-образной модели** определяется четыре базовых шага :

1. ***анализ*** (планирование проекта и определение системных требований);
2. ***проектирование*** (разделяется на высокоуровневое – определение структуры программного продукта, взаимосвязи между основными его компонентами и реализуемых ими функций – и низкоуровневое (детальное) – определение работы каждого компонента);
3. ***разработка*** (кодирование) – преобразование алгоритмов в готовое программное обеспечение;
4. ***обзор*** (различные виды тестирования):

- *Модульное тестирование* – проверка каждого компонента или модуля программного продукта (ПП).
- *Интеграционное тестирование* – интеграция ПП и его тестирование.
- *Системное тестирование* – проверка функционирования ПП после помещения его в аппаратную среду в соответствии со спецификацией требований.
- *Эксплуатация и сопровождение* – запуск программного продукта в производство, внесение поправок и модернизация ПП.

2. Обзор моделей жизненного цикла программного обеспечения

V-образную модель целесообразно использовать при разработке программных продуктов, для которых главным требованием является **надежность**.

Преимущества	Недостатки
• Большая роль придается верификации и аттестации программного продукта, начиная с ранних стадий его разработки. • Все действия планируются. • Предполагаются аттестации и верификации как самого программного продукта, так и всех полученных внутренних и внешних данных. • Завершение каждой фазы является контрольной точкой, поэтому ход выполнения работы легко отслеживается	• Не учитываются итерации между фазами. • Нельзя вносить изменения на разных этапах жизненного цикла. • Тестирование требований происходит очень поздно, в результате внесение изменений влияет на выполнение графика работ

2. Обзор моделей жизненного цикла программного обеспечения

Когда использовать V-модель?

- Если требуется тщательное тестирование продукта, то V-модель оправдает заложенную в себя идею: validation and verification.
- Для малых и средних проектов, где требования четко определены и фиксированы.
- В условиях доступности инженеров необходимой квалификации, особенно тестировщиков.

2. Обзор моделей жизненного цикла программного обеспечения

Модель прототипирования позволяет создать прототип программного продукта до или в течение этапа составления требований к нему. С этим прототипом работают потенциальные пользователи, они определяют его сильные и слабые стороны и сообщают о результатах разработчикам ПП.

2. Обзор моделей жизненного цикла программного обеспечения

Модели прототипирования рекомендуется применять в случаях, когда:

- требования к программному продукту заранее не известны;
- требования не постоянны или неудачно сформулированы, их необходимо уточнять;
- необходима проверка концепции;
- существует потребность в пользовательском интерфейсе;
- выполняется новая, не имеющая аналогов разработка;
- разработчики не уверены, какое решение следует выбрать.

2. Обзор моделей жизненного цикла программного обеспечения

Модель прототипирования

Преимущества	Недостатки
• Благодаря реакции заказчика на прототип число неточночтей в требованиях сводится к минимуму. • Снижается вероятность возникновения путаницы, искажения информации и недоразумений при определении требований к продукту. В результате создается более качественный программный продукт. • В процессе разработки можно учесть новые требования заказчика. • Прототип позволяет очень гибко выполнять проектирование и разработку. • Заказчик видит прогресс в процессе разработки программного продукта. • Сведена к минимуму возможность возникновения противоречий между заказчиками и разработчиками. • Уменьшается число доработок и, как следствие, стоимость разработки. • Возникающие проблемы решаются на ранних стадиях, что позволяет сократить расходы на их устранение	• Решение сложных задач может отодвигаться на будущее. • Заказчик может предпочесть получить прототип, а не законченную версию программного продукта. • Пототипирование может неоправданно затянуться. • До начала работ неизвестно, сколько итераций придется выполнить

2. Обзор моделей жизненного цикла программного обеспечения

Классификация прототипов:

- **Горизонтальные** прототипы — моделирует исключительно UI не затрагивая логику обработки и базу данных.
- **Вертикальные** прототипы — проверка архитектурных решений.
- **Одноразовые** прототипы — для быстрой разработки.
- **Эволюционные** прототипы — предварительная реализация ПП.

2. Обзор моделей жизненного цикла программного обеспечения

Многопроходная модель представляет собой несколько итераций процесса построения прототипа программного продукта с добавлением на каждой итерации новых функциональных возможностей или повышением эффективности продукта.

Инкремент – процесс добавления к программному продукту новых функциональных возможностей и относящихся к ним функций.

2. Обзор моделей жизненного цикла программного обеспечения

Многопроходная модель

Преимущества	Недостатки
• В начале разработки требуются средства только для разработки и реализации основных функций продукта. • После каждого инкремента получается функциональный продукт. • Снижается риск неудачи и изменения требований. • Улучшается понимание требований для поздних итераций как разработчиками, так и пользователями. • Инкременты функциональных возможностей легко поддаются тестированию	• Не предусмотрены итерации внутри каждого инкремента. • Определение полной функциональности должно осуществляться в самом начале жизненного цикла разработки. • Может возникнуть тенденция откладывания решения трудных задач. • Обязательно наличие хорошего планирования и проектирования

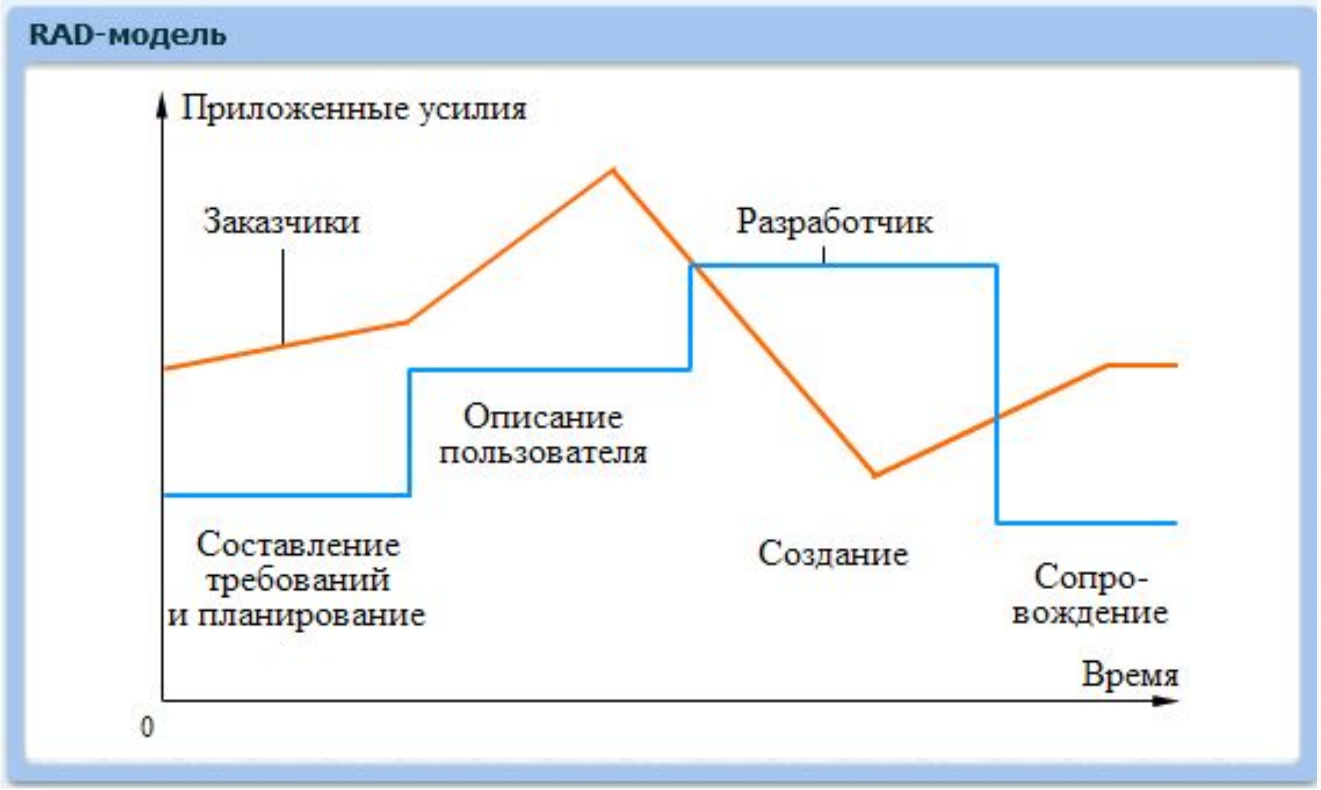
Когда использовать многопроходную модель?

•большинство требований к продукту будут сформулированы заранее;

•для выполнения проекта отведено много времени.

2. Обзор моделей жизненного цикла программного обеспечения

RAD – модель (Rapid Application Development - модель быстрой разработки) - решающую роль играет конечный пользователь, участвующий совместно с разработчиками в формировании требований и апробации их на работающих прототипах.



2. Обзор моделей жизненного цикла программного обеспечения

RAD-модель — разновидность инкрементной модели. В RAD-модели компоненты или функции разрабатываются несколькими высококвалифицированными командами параллельно, будто несколько мини-проектов. Временные рамки одного цикла жестко ограничены. Созданные модули затем интегрируются в один рабочий прототип. Синергия позволяет очень быстро предоставить клиенту для обозрения что-то рабочее с целью получения обратной связи и внесения изменений.

1. Обзор моделей жизненного цикла программного обеспечения

RAD-модель включает следующие фазы:

1. **Бизнес-моделирование:** определение списка информационных потоков между различными подразделениями.
2. **Моделирование данных:** информация, собранная на предыдущем этапе, используется для определения объектов и иных сущностей, необходимых для циркуляции информации.
3. **Моделирование процесса:** информационные потоки связывают объекты для достижения целей разработки.
4. **Сборка приложения:** используются средства автоматической сборки для преобразования моделей системы автоматического проектирования в код.
5. **Тестирование:** тестируются новые компоненты и интерфейсы.

2. Обзор моделей жизненного цикла программного обеспечения

RAD-модель

Преимущества

- Использование современных инструментальных средств разработки позволяет сократить время цикла разработки.
- Привлечение к работе заказчика сводит к минимуму риск, что он останется недоволен готовым продуктом.
- Повторно используются компоненты уже существующих программ

Недостатки

- Если заказчики не могут постоянно участвовать в процессе разработки, это может негативно сказаться на продукте.
- Для работы требуются высококвалифицированные кадры, владеющие современными инструментальными средствами.
- Работа над продуктом может быть зациклена, поэтому существует риск, что она никогда не будет завершена (важно вовремя остановиться)

Подходит, если:

- важна скорость и простота разработки;
- четко определены приоритетные направления разработки;
- сжатые сроки разработки;
- ограниченный бюджет;
- главный критерий – интерфейс пользователя;
- есть возможность разбить проект на функциональные компоненты.

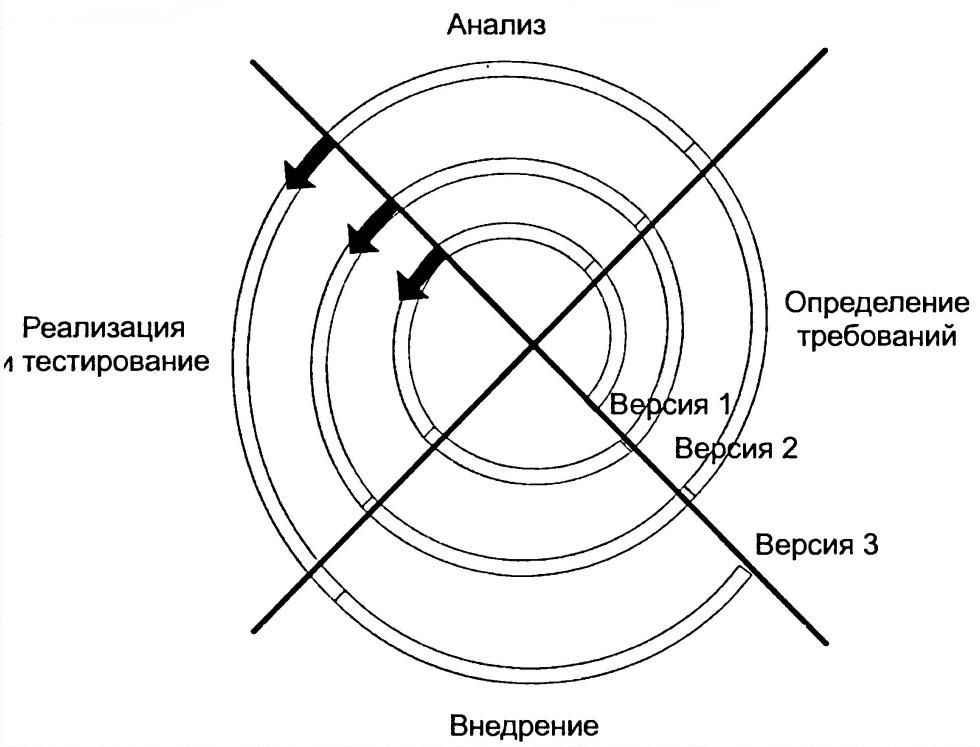
Не подходит, если

- в приоритете качество и контроль;
- крупномасштабный проект;
- критически важен высокий уровень планирования и жесткая дисциплина проектирования;
- от приложения в определенной степени зависит безопасность людей.

2. Обзор моделей жизненного цикла программного обеспечения

Спиральная модель жизненного цикла – поддерживает итерации поэтапной модели, но особое внимание уделяется начальным этапам проектирования:

- анализу требований, проектированию спецификаций,
- предварительному проектированию и детальному проектированию.

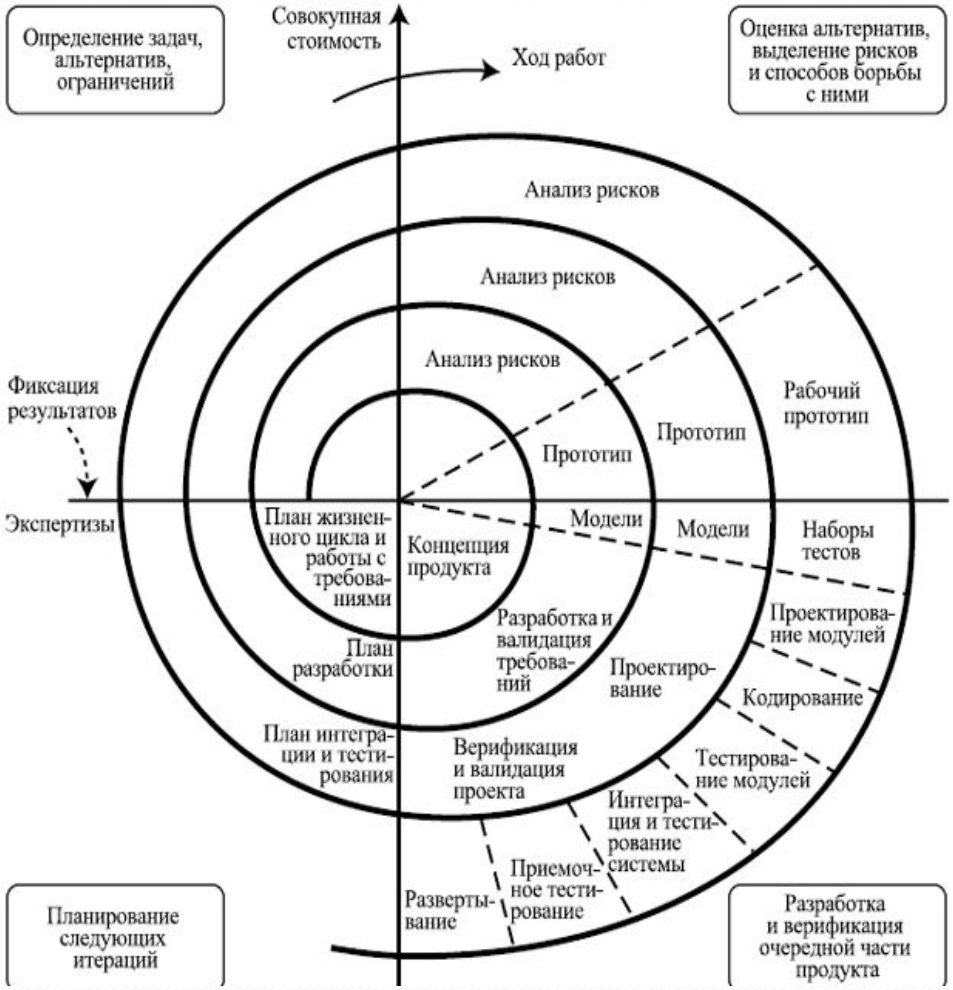


Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии ПО, уточняются цели и требования к программному обеспечению, оценивается качество разработанного фрагмента или версии и планируются работы следующей стадии разработки (витка).

2. Обзор моделей жизненного цикла программного обеспечения

Основные особенности спиральной модели:

- прикладной программный продукт создается по частям с использованием метода прототипирования;
- создание прототипов осуществляется за несколько итераций (витков спирали);
- каждая итерация соответствует созданию фрагмента или версии ПП, на ней уточняются цели и характеристики проекта, оценивается качество полученных результатов, планируются работы следующей итерации.



2. Обзор моделей жизненного цикла программного обеспечения

Разработка итерациями позволяет переходить на следующую стадию до полного завершения работ на текущей, поскольку недостающую работу можно выполнить на следующей итерации.

Основная проблема: спирального цикла является определение момента перехода на следующую стадию. **Решение:** вводятся временны́е рамки для каждой из стадий жизненного цикла.

Преимущества	Недостатки
• Заказчик может увидеть разрабатываемый программный продукт на ранних стадиях разработки. • В разработке продукта заказчики принимают активное участие. • Модель сочетает преимущества каскадной и многопроходной моделей	• Усложненная структура. • Спираль может продолжаться до бесконечности: каждая ответная реакция заказчика может породить новый цикл

Когда использовать спиральную модель?

- целесообразно создание прототипа;
- требуется выполнять проекты со средней и высокой степенями риска;
- заказчики не уверены в своих потребностях;
- требования слишком сложные;
- проект очень большой.

2. Обзор моделей жизненного цикла программного обеспечения

Rational Objectory Process (ROP)

Объектно-ориентированное проектирование ПО стало результатом появления объектно-ориентированного программирования (ООП). Фирма Rational Software, разработавшая язык UML, предложила также и свою модель жизненного цикла - **Rational Objectory Process** (ROP).

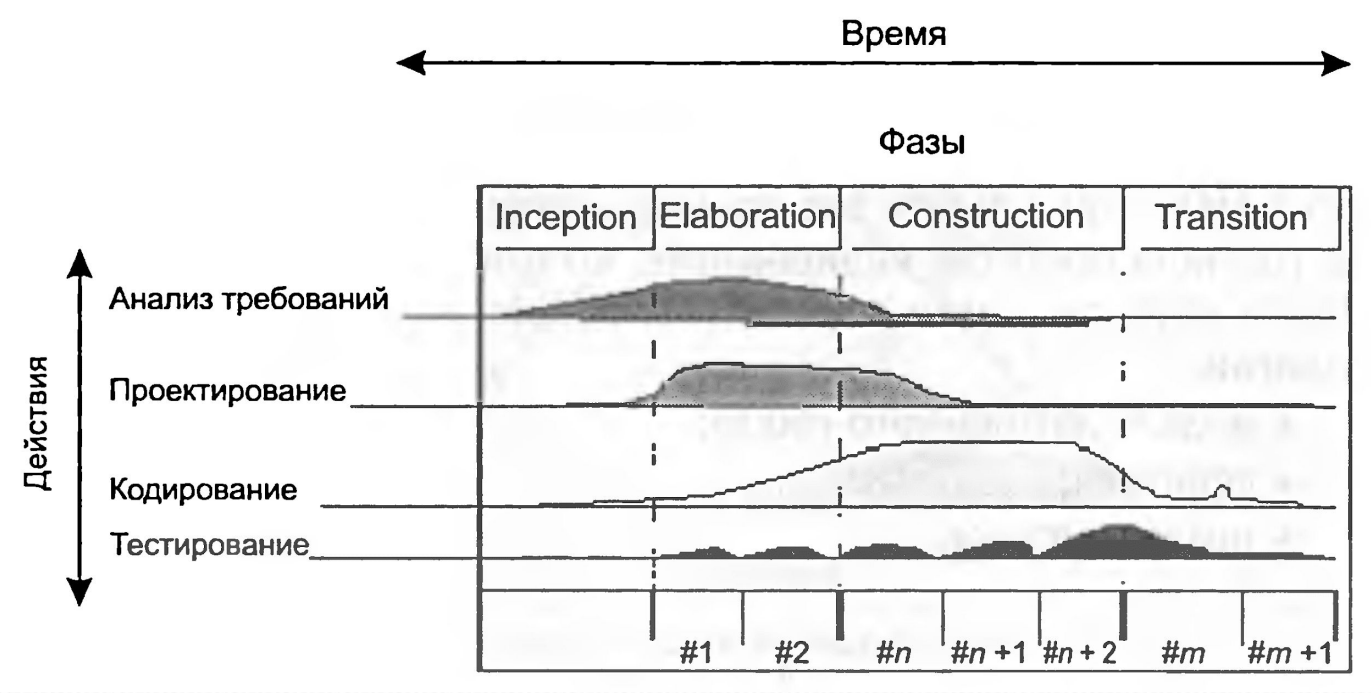
Основные свойства ROP-технологии:

- Rational Objectory Process – итеративный процесс, в течение которого происходит последовательное уточнение результатов.
- Rational Objectory Process направлен именно на создание моделей, а не на разработку каких-либо других элементов проекта (например, текстовых документов).
- Действия Rational Objectory Process определяются в первую очередь блоками использования (use case).

2. Обзор моделей жизненного цикла программного обеспечения

ROP разбит на циклы, каждый из которых состоит из четырех фаз:

- начальная стадия (Inception);
- разработка (Elaboration);
- конструирование (Construction);
- ввод в эксплуатацию (Transition).



2. Обзор моделей жизненного цикла программного обеспечения

Начальная стадия - начальный анализ оценки проекта. Предполагает

- всестороннее изучение всех возможностей реализации (для крупных проектов);
- выработку бизнес-плана проекта;
- определение его стоимости, примерный доход;
- ограничения ресурсов.

Окончанием начального этапа могут служить следующие результаты:

- начальный проектный словарь терминов;
- общее описание системы – основные требования к проекту, его характеристики и ограничения;
- начальная модель вариантов использования;
- начальный бизнес-план;
- план проекта, отражающий стадии и итерации;
- один или несколько прототипов.

2. Обзор моделей жизненного цикла программного обеспечения

На ***стадии разработки*** выявляются более детальные требования к системе, выполняется высокоуровневый анализ предметной области и проектирование базовой архитектуры системы, создается план конструирования и устраняются наиболее рискованные элементы проекта.

Результаты:

1) ***описание базовой архитектуры*** будущей системы, которая включает:

• модель предметной области;

• технологическую платформу, определяющую основные элементы технологии реализации системы и их взаимодействие.

2) о***ценка времени реализации*** *каждого* ***варианта*** *использования*;

•3) ***идентификация*** всех наиболее серьезных ***рисков и возможности их ликвидации***.

2. Обзор моделей жизненного цикла программного обеспечения

Сущность ***стадии конструирования*** заключается в определении последовательности итераций конструирования и вариантов использования, реализуемых на каждой итерации, которые являются одновременно инкрементными и повторяющимися.

Особенности:

• итерации являются инкрементными в соответствии с выполняемой функцией;

• итерации являются повторяющимися по отношению к разрабатываемому коду.

Результатом стадии конструирования является продукт, готовый к передаче пользователям и содержащий, как правило, руководство пользователей и готовый к интеграции на требуемых платформах.

2. Обзор моделей жизненного цикла программного обеспечения

Назначением ***стадии ввода в эксплуатацию*** является передача готового продукта в полное распоряжение конечных пользователей.

Данная стадия включает:

• бета-тестирование, позволяющее убедиться, что новая система соответствует ожиданиям пользователей;

• параллельное функционирование с существующей (legacy) системой, которая подлежит постепенной замене;

• оптимизацию производительности;

• обучение пользователей и специалистов службы сопровождения.

*__Бета-тестирование__ – интенсивное использование почти готовой версии программного продукта с целью выявления максимального числа ошибок в его работе для их последующего устранения.

Альфа-тестирование – имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальным заказчиком, чаще всего проводимая на ранней стадии разработки программного продукта.

3. Процессы жизненного цикла программного обеспечения

В соответствии со стандартом, работы, которые могут выполняться в жизненном цикле программного средства, распределены по следующим процессам **ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств»**:

3. Процессы жизненного цикла программного обеспечения

Основные процессы:

1. *Процесс заказа*. Определяет работы заказчика, то есть организации, которая приобретает программный продукт.

Данный процесс состоит из следующих работ:

1)подготовка;
2)подготовка заявки на подряд;
3)подготовка и корректировка договора;
4)надзор за поставщиком;
5)приемка и закрытие договора.

2. *Процесс поставки*. Определяет работы поставщика, то есть организации, которая поставляет программный продукт.

Данный процесс состоит из следующих работ:

1)подготовка;
2)подготовка ответа;
- подготовка договора;

1)планирование;
2)выполнение и контроль;
3)проверка и оценка;
4)поставка и закрытие договора.

3. Процессы жизненного цикла программного обеспечения

Основные процессы:

3. *Процесс разработки*. Определяет работы разработчика, то есть организации, которая проектирует и разрабатывает программный продукт.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)анализ требований к системе;
3)проектирование системной архитектуры;
4)анализ требований к программным средствам;
5)проектирование программной архитектуры;
6)техническое проектирование программных средств;
7)программирование и тестирование программных средств;

- сборка программных средств;

1)квалификационные испытания программных средств;
2)сборка системы;
3)квалификационные испытания системы;
4)ввод в действие программных средств;
5)обеспечение приемки программных средств.

3. Процессы жизненного цикла программного обеспечения

Основные процессы:

4. *Процесс эксплуатации*. Определяет работы оператора, то есть организации, которая обеспечивает эксплуатационное обслуживание вычислительной системы в данных условиях в интересах пользователей.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)эксплуатационные испытания;
3)эксплуатация системы;
4)поддержка пользователя.

5. *Процесс сопровождения*. Определяет работы персонала сопровождения, то есть организации, которая производит контролируемые изменения программного продукта с целью сохранения его исходного состояния и функциональных возможностей.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)анализ проблем и изменений;
- внесение изменений;
- проверка и приемка при сопровождении;

1)перенос;
2)снятие с эксплуатации.

3. Процессы жизненного цикла программного обеспечения

Вспомогательные процессы:

1. *Документирование* – процесс формализованного описания информации, созданной в процессе или работе жизненного цикла.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)проектирование и разработка;
3)выпуск;
4)сопровождение.

Документирование заканчивается на этапе реализации.

2. *Управление конфигурацией* – процесс применения различных процедур на всем протяжении жизненного цикла программных средств для управления программными объектами в системе.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)определение конфигурации;
3)контроль конфигурации;
•учет состояний конфигурации;
1)оценка конфигурации;
2)управление выпуском и поставка.

3. Процессы жизненного цикла программного обеспечения

Вспомогательные процессы:

3. *Обеспечение качества* – процесс обеспечения гарантий того, что программные продукты и процессы в жизненном цикле соответствуют установленным требованиям и утвержденным планам.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)обеспечение продукта;
3)обеспечение процесса;
4)обеспечение систем качества.

4. *Верификация* – процесс определения того, что программные продукты функционируют в полном соответствии с требованиями или условиями, реализованными в предшествующих работах.

5. *Аттестация* – процесс определения полноты соответствия установленных требований, созданной системы или программного продукта их функциональному назначению.

3. Процессы жизненного цикла программного обеспечения

Вспомогательные процессы:

6. *Совместный анализ* – процесс оценки состояний и результатов работ по проекту. Данный процесс может выполняться двумя любыми сторонами, участвующими в договоре.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)анализы управления проектом;
3)технические анализы.

7. *Аудит* – процесс определения соответствия требованиям, планам и условиям договора. Данный процесс может выполняться двумя любыми сторонами, участвующими в договоре.

Данный процесс состоит из следующих работ:

1)подготовка процесса;
2)аудиторская проверка.

8. *Решение проблем*. Процессы жизненного цикла ПО с точки зрения вовлеченных в него сторон. – процесс анализа и решения проблем, которые обнаружены в ходе выполнения разработки, эксплуатации, сопровождения или других процессов.

3. Процессы жизненного цикла программного обеспечения

Организационные процессы:

1. ***Процесс управления*** – общие работы и задачи, которые могут быть использованы любой стороной, управляющей соответствующим процессом.

2. ***Процесс создания инфраструктуры*** – процессом установления и обеспечения инфраструктуры, необходимой для любого другого процесса.

3. ***Процесс усовершенствования*** – процессом установления, оценки, измерения, контроля и улучшения любого процесса жизненного цикла программных средств.

4. ***Процесс обучения*** – процесс обеспечения первоначального и продолженного обучения персонала.

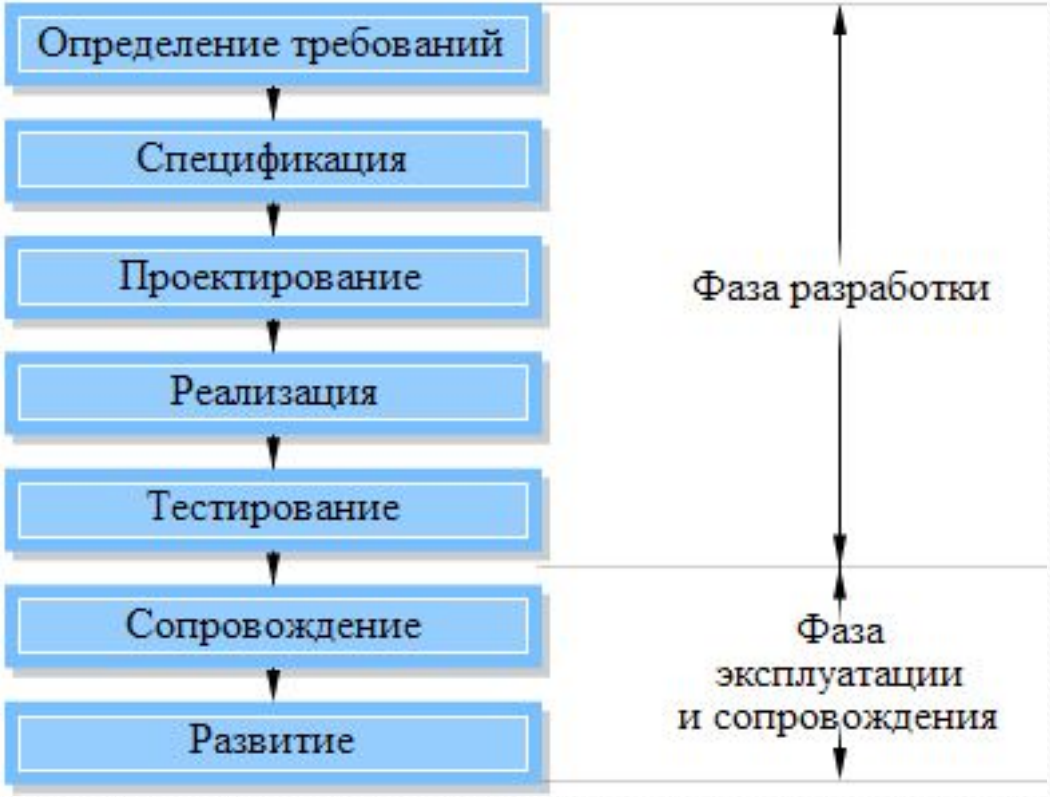
3. Процессы жизненного цикла программного обеспечения

Процессы жизненного цикла ПО с точки зрения вовлеченных в него сторон

Заказчик Поставщик	Контрактный вид	Процесс заказа Процесс поставки	
Оператор Пользователь	Операционный вид	Процесс функционирования	
Разработчик Поддержка	Инженерный вид	Процесс поддержки	Процесс разработки
Сотрудники поддержки	Поддерживающий вид	Поддерживающие процессы • документирование • управление конфигурацией • обеспечение качества • аттестация • совместный обзор • аудит • разрешение проблем	
Менеджер	Корпоративный вид	Организационные процессы • менеджмент • инфракструктура • улучшение • обучение	

3. Процессы жизненного цикла программного обеспечения

Общепринятая модель жизненного цикла программного обеспечения



Этап – часть процесса создания программного продукта, ограниченная определенными временными рамками и заканчивающаяся выпуском конкретного продукта (моделей, программных компонентов, документации), определяемого заданными для данного этапа требованиями.)

4. Единая система программной документации (ЕСПД)

Единая система программной документации (ЕСПД) — комплекс государственных стандартов Российской Федерации, устанавливающих взаимосвязанные правила разработки, оформления и обращения программ и программной документации.

Код группы	Наименование группы
0	Общие положения
1	Основополагающие стандарты
2	Правила выполнения документации разработки
3	Правила выполнения документации изготовления
4	Правила выполнения документации сопровождения
5	Правила выполнения эксплуатационной документации
6	Правила обращения программной документации
7	Резервные группы
8	Прочие стандарты

ГОСТ 19.101-77 Виды программ и программных документов

4. Единая система программной документации (ЕСПД)

К **программным** относят документы, содержащие сведения, необходимые для *разработки*, *сопровождения* и *эксплуатации* ПО.

- Спецификация должна содержать перечень и краткое описание назначения всех файлов программного обеспечения, в том числе и файлов документации на него, и является обязательной для программных систем, а также их компонентов, имеющих самостоятельное применение.
- Ведомость держателей подлинников должна содержать список предприятий, на которых хранятся подлинники программных документов. Необходимость этого документа определяется на этапе разработки и утверждения технического задания только для программного обеспечения со сложной архитектурой.
- Текст программы должен содержать текст программы с необходимыми комментариями. Необходимость этого документа определяется на этапе разработки и утверждения технического задания.

4. Единая система программной документации (ЕСПД)

- Описание программы должно содержать сведения о логической структуре и функционировании программы.
- Программа и методика испытаний должны содержать требования, подлежащие проверке при испытании программного обеспечения, а также порядок и методы их контроля.
- Техническое задание должно содержать сведения о назначении и области применения программы, технических, технико-экономических и специальных требованиях, предъявляемых к программе, необходимых стадиях и сроках разработки, видах испытаний.
- Пояснительная записка должна содержать информацию о структуре и конкретных компонентах программного обеспечения, в том числе схемы алгоритмов, их общее описание, а также обоснование принятых технических и технико-экономических решений. Составляется на стадии эскизного и технического проектов.
- Эксплуатационные документы должны содержать сведения для обеспечения функционирования и эксплуатации программы.

4. Единая система программной документации (ЕСПД)

ГОСТ 19.102-77 Стадии разработки

Стадии разработки	Этапы работ	Содержание работ
Технический проект	Разработка технического проекта	Уточнение структуры входных и выходных данных. Разработка алгоритма решения задачи. Определение формы представления входных и выходных данных. Определение семантики и синтаксиса языка. Разработка структуры программы. Окончательное определение конфигурации технических средств.
	Утверждение технического проекта	Разработка плана мероприятий по разработке и внедрению программ Разработка пояснительной записки. Согласование и утверждение технического проекта.
Рабочий проект	Разработка программы	Программирование и отладка программы
	Разработка программной документации	Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.
	Испытания программы	Разработка, согласование и утверждение программы и методики испытаний. Проведение предварительных государственных, межведомственных, приемо-сдаточных и других видов испытаний. Корректировка программы и программной документации по результатам испытаний.
Внедрение	Подготовка и передача программы	Подготовка и передача программы и программной документации для сопровождения и (или) изготовления Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление. Передача программы в фонд алгоритмов и программ

4. Единая система программной документации (ЕСПД)

ГОСТ 19.105-78 Общие требования к программным документам

ГОСТ 19.104-78 Основные надписи

ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.202-78 Спецификация. Требования к содержанию и оформлению

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению

4. Единая система программной документации (ЕСПД)

Вид программы	Определение
Компонент	Программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса
Комплекс	Программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции, и применяемая самостоятельно или в составе другого комплекса

Выводы

- В зависимости от специфики разрабатываемого программного продукта, возможностей разработчика, требований заказчика к тем или иным программным продуктам применяются различные модели жизненного цикла.
- Правильный выбор модели жизненного цикла позволяет оптимизировать процесс разработки программного продукта и повысить его качество.
- Требования к составу, содержанию и оформлению документов, описывающих программу на разных стадиях ее жизненного цикла содержатся в комплексе государственных стандартов РФ, составляющих **Единую систему программной документации (ЕСПД)**