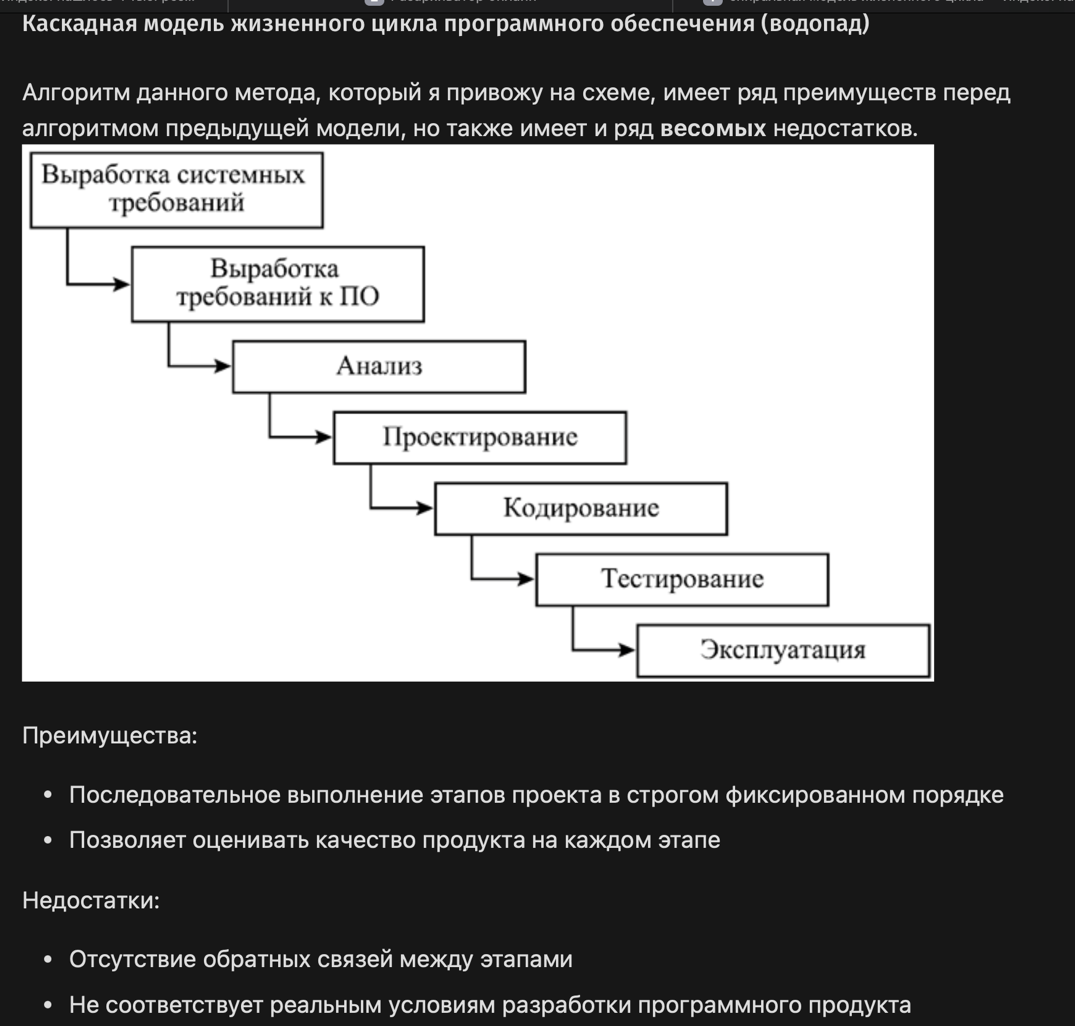
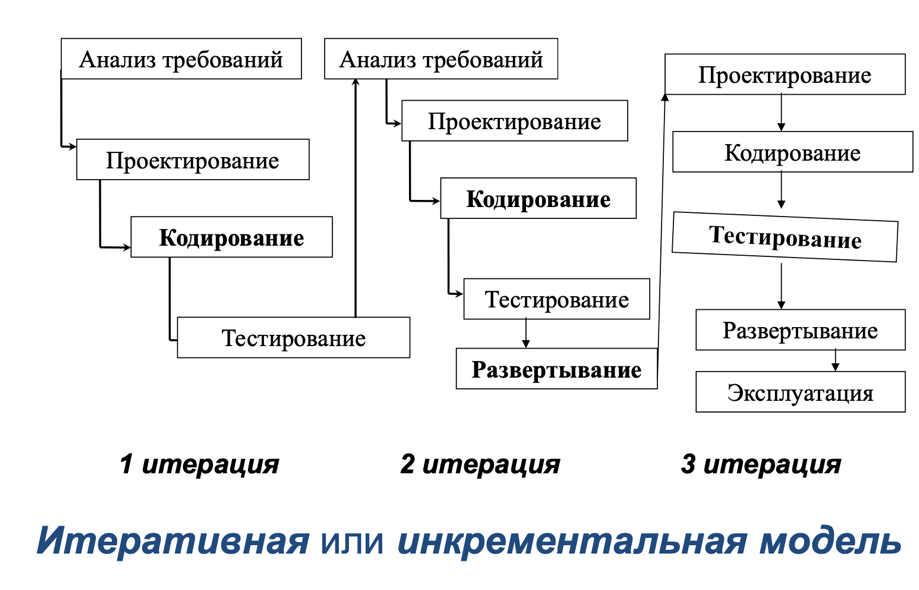
1. **Понятие жизненного цикла программных средств. Модели жизненного цикла.**

Жизненным циклом ПО называют весь период существования ПО, связанный с подготовкой к его разработке, разработкой, использованием и переработками, начиная с того момента, когда принимается решение о его разработке до того момента, когда полностью прекращается всякое ее использование

**Каскадная**



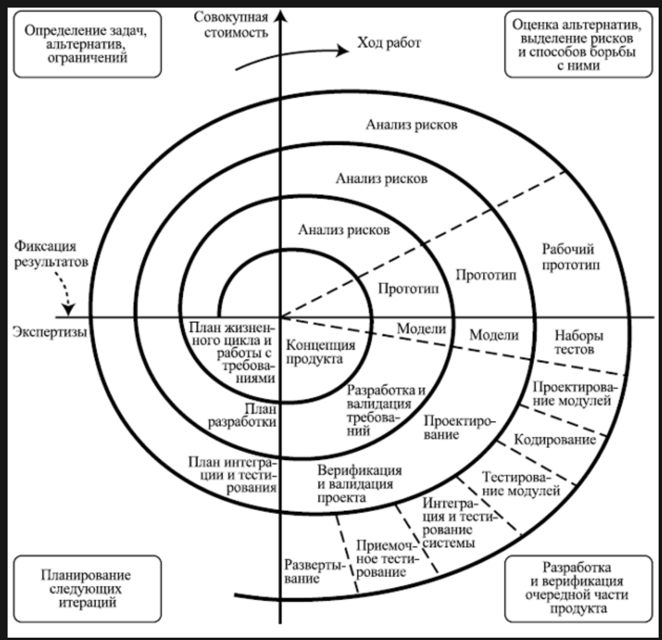
**Итеративная модель** предполагает разбиение жизненного цикла проекта на последовательность итераций, каждая из которых напоминает «минипроект»



**Спиральная**

Каждый виток спирали соответствует созданию фрагмента или версии программного обеспечения, на нём уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Каждый виток разбит на 4 сектора:

* определение целей,
* оценка и разрешение рисков,
* разработка и тестирование,
* планирование следующей итерации.



Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла. Боэм формулирует десять наиболее распространённых (по приоритетам) рисков:

* Дефицит специалистов.
* Нереалистичные сроки и бюджет.
* Реализация несоответствующей функциональности.
* Разработка неправильного пользовательского интерфейса.
* «Золотая сервировка», перфекционизм, ненужная оптимизация и оттачивание деталей.
* Непрекращающийся поток изменений.
* Нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию.
* Недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами.
* Недостаточная производительность получаемой системы.
* Разрыв между квалификацией специалистов и требованиями проекта[1]

1. **Методы тестирования программных систем. Процессы и средства тестирования программных компонентов.**

Тестирование ПС от требований имеет две взаимодополняющие цели:

* Показать, что ПС удовлетворяет заданным требованиям к нему.
* Показать с высокой степенью доверия, что устранены дефекты и ошибки, которые могли бы привести к возникновению отказов, влияющих на корректность и надежность системы.

При первой стратегии за основу принимается структура ПМ, построенная по тексту программы в виде графа (УГП).

1. В УПГ выделяются и упорядочиваются маршруты исполнения программ.
2. Эти условия используются для подготовки тестовых наборов, каждый из которых должен реализоваться по маршруту, принятому за эталон при подготовке теста.
3. Отклонение исполнения теста от первоначально выбранного маршрута рассматривается как ошибка

При второй стратегии за основу принимаются требования спецификаций, конкретные тестовые и эталонные значения.

1. При каждом тесте программа исполняется по определенному маршруту, который регистрируется.
2. По мере тестирования отмечаются проверенные операторы, и оценивается полнота покрытия тестами требований спецификаций на маршрутах тестирования

Программы в процессе тестирования используются в двух различных

формах представления:

1. текст программы на языке программирования или формализованного описания спецификаций требований (графическое представление);

2. Текст в машинном коде конкретной ЭВМ (объектное представление).

Системы автоматизации тестирования можно разделить на два вида:

* статические системы, которые анализируют спецификации и исходные тексты программ в объектном коде без их исполнения на ЭВМ (анализаторы);
* Динамические системы, при использовании которых программы функционируют в объектном коде и пригодны для их реального применения на ЭВМ.

В группу средств статической отладки входят также средства расчета длительностей исполнения модулей и компонентов. В результате расчетов выявляются компоненты программы, требующие избыточно большого времени счета на реализующей ЭВМ.

Группу средств динамической отладки можно разделить на два типа:

1. Основные средства, непосредственно обеспечивающие исполнение программ в соответствии с отладочными заданиями (транслятор, исполнение по сценарию, регистрация результатов).

2. Вспомогательные средства, которые анализируют выполненное тестирование, его результаты и проведенные корректировки.

1. **Проблемы современной программной инженерии. Понятие сложной системы. Типы сложных систем.**

Сложность анализируемых объектов – комплексов программ и психологическая самоуверенность ряда программистов в собственной “непогрешимости”, часто приводят к тому, что реальные характеристики качества функционирования программных продуктов остаются неизвестными не только для заказчиков и пользователей, но также для самих разработчиков.

Отсутствие четкого декларирования в документах понятий и требуемых значений характеристик качества ПС вызывает конфликты между заказчиками-пользователями и разработчиками-поставщиками из-за разной трактовки одних и тех же характеристик.

Методология программной инженерии и стандарты регламентируют современные процессы управления проектами сложных систем и программных средств.

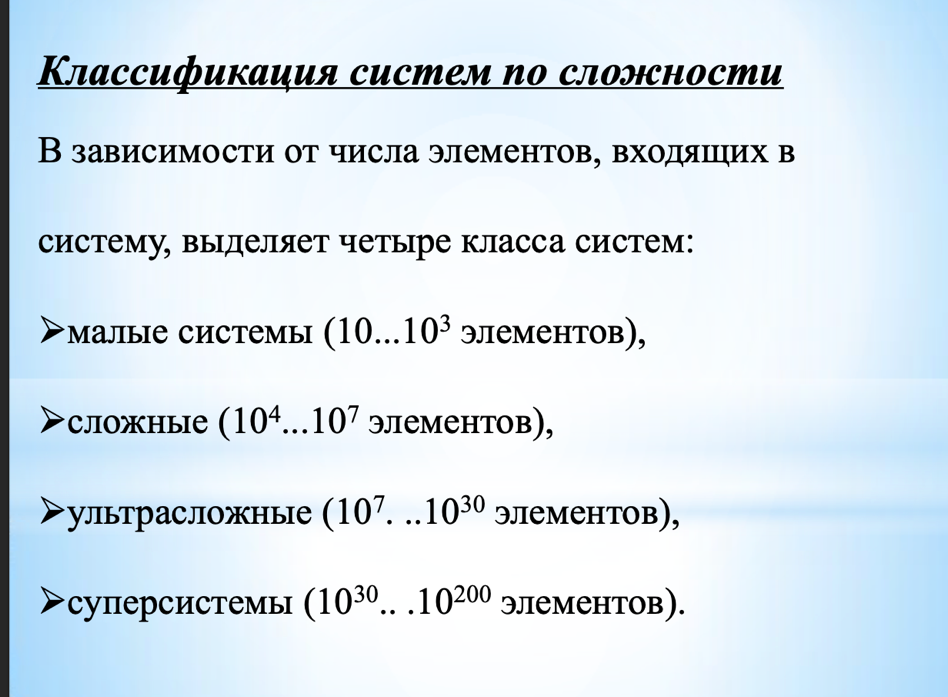
**Практическое применение профилей стандартов, сосредоточивших мировой опыт создания различных типов крупных комплексов программ, способствует значительному повышению производительности труда специалистов и качества создаваемых программных продуктов.**

*1. Система, которая разрабатывается не одним человеком, а группой разработчиков (более 5 человек).*

*2.Если количество строк исходного кода исчисляется сотнями тысяч или даже миллионами.*

*3.Если сложную задачу можно декомпозировать на более простые задачи (которые будут реализованы более мелкими подсистемами).*

*4.Если в требованиях встречаются взаимоисключающие требования. Например, нужно обработать огромные информационные потоки, и время отклика должно быть минимальным. Следует найти компромисс.*

* Она решает одну или несколько связанных задач,
* Она должна быть удобной в использовании.
* Должна включать достаточно полную и понятную пользователям документацию
* Должна включать набор документов для обучения пользователей работе с программой.
* Ее низкая производительность на реальных данных приводит к значимым потерям для пользователей.
* Ее неправильная работа наносит ощутимый ущерб пользователям и другим организациям и лицам, даже если сбои происходят не слишком часто.
* Для выполнения своих задач она должна взаимодействовать с другими программами и программно-аппаратными системами, работать на разных платформах
* Велико количество ее возможных пользователей.

1. **Интеграция, квалификационное тестирование и испытания комплексов программ. Подходы для интеграционного тестирования комплексов программ.**

Квалификационное тестирование -- тестирование, проводимое разработчиком и санкционированное приобретающей стороной (при необходимости) с целью демонстрации того, что программный продукт удовлетворяет спецификациям и готов для применения в заданном окружении или интеграции с системой, для которой он предназначен.

Интеграционное тестирование комплексов программ - это процесс проверки взаимодействия между различными компонентами программного комплекса для обеспечения их корректной работы вместе.

**Подходы для интеграционного тестирования комплексов программ:**

- Сверху-вниз (Top-Down): При этом подходе тестирование начинается с верхних уровней системы и постепенно опускается к нижним. Это позволяет выявить проблемы взаимодействия между компонентами раньше, чем если бы тестирование начиналось с нижних уровней.

- Снизу-вверх (Bottom-Up): В отличие от верхнего подхода, здесь тестирование начинается с низших уровней системы и постепенно продвигается к верхним. Это позволяет выявить проблемы на уровне компонентов и модулей до их интеграции в систему.

- Покомпонентный (Component Stubs): Здесь тестируются отдельные компоненты системы с использованием заглушек (stubs) для имитации поведения других компонентов. Это позволяет проверить работоспособность отдельных частей системы независимо от других компонентов.

1. **Профили стандартов жизненного цикла систем и программных средств в области программной инженерии.**
2. ISO/IEC 12207 «Standard for Information Technology Software Life Cycle Processes»(1995) -- Определяет общую структуру жизненного цикла ПО в виде 3-х ступенчатой модели, состоящей из процессов, видов деятельности и задач.
3. Стандарт ISO 15846 обобщает, детализирует и развивает основные концептуальные положения в части управления конфигурацией, представленные в стандарте ISO 12207.
4. Процесс сопровождения ПС проводится в соответствии со стандартом ISO 12207 и детализацией этого раздела в стандарте ISO 14764.
5. Стандарт ISO 15504:1-5:2003-2006 регламентирует оценку и аттестацию зрелости процессов создания, сопровождения и совершенствования программных средств и систем, выполняемых предприятиями (из лекции про CMM, CMMI, POSIX). Применение стандарта ISO 15504:1-5:2003-2006 направлено на выработку предприятиями и специалистами культуры постоянного совершенствования зрелости технологий обеспечения ЖЦ ПС, отвечающих бизнес-целям проектов и оптимизации использования доступных ресурсов.
6. Серия стандартов ISO 9000:2000 разработана, чтобы помочь предприятиям всех типов и размеров внедрить и использовать эффективные системы менеджмента (административного управления) качества.
7. Новую версию международных стандартов POSIX составляет стандарт ISO 9945:1-4:2003 – ИТ + Стандарт ISO 14252:1996 -- Руководство по POSIX. Стандарт POSIX (Portable Operating System Interface) представляет собой набор стандартов, разработанных для обеспечения совместимости между различными операционными системами. Он определяет интерфейсы между операционной системой и прикладным программным обеспечением, что позволяет приложениям быть переносимыми между различными UNIX-подобными операционными системами.
8. Стандарты качества ПО:

* ISO 9126:1991. (ГОСТ РФ – 1993). ИТ. Оценка программного продукта. Характеристики качества и руководство по их применению.
* ISO 9126-1-4: 2002. ИТ. ТО. Качество программных средств: Ч.1.

1. Стандарт СТО БР ИББС – 1.0-2010 определяет модель угроз информационной безопасности
2. **Процессы эксплуатации и сопровождения ПС в жизненном цикле. Этапы и процедуры при сопровождении программных средств в соответствии с требованиями стандарта ISO 12207 по развитию и модификации программного продукта в жизненном цикле.**

Процессы эксплуатации и сопровождения ПС в жизненном цикле

Жизненный цикл программного средства (ПС) включает в себя несколько этапов, одним из которых является эксплуатация и сопровождение. Эти процессы направлены на обеспечение надёжной и эффективной работы программного продукта после его внедрения.

Эксплуатация ПС — это процесс использования программного продукта пользователями для выполнения своих задач. В ходе эксплуатации могут возникать различные проблемы, связанные с работой программы, которые необходимо решать.

В соответствии с требованиями стандарта ISO 12207 (ISO 14764) по развитию и модификации программного продукта в ЖЦ должен быть организован процесс его сопровождения.

основная задача сопровождения - изменить и улучшить существующий программный продукт, сохраняя его целостность и функциональную пригодность.

Работы по сопровождению ПС:

\*подготовка процесса;

\*анализ проблем и изменений;

\*внесение изменений;

\*проверка и приемка при сопровождении;

\*перенос на иные платформы;

\*снятие с эксплуатации.

Общий план сопровождения должен определять:

- причины сопровождения;

- состав исполнителей работ;

- роли и обязанности каждого субъекта, вовлеченного в сопровождение;

- как должны быть выполнены основные процессы;

- какие имеются и какие необходимы ресурсы для сопровождения;

- методы и средства организации работ по управлению, выпуску продукта и синхронизации работ;

- перечень всех проектных результатов и продуктов, подлежащих поставке заказчику;

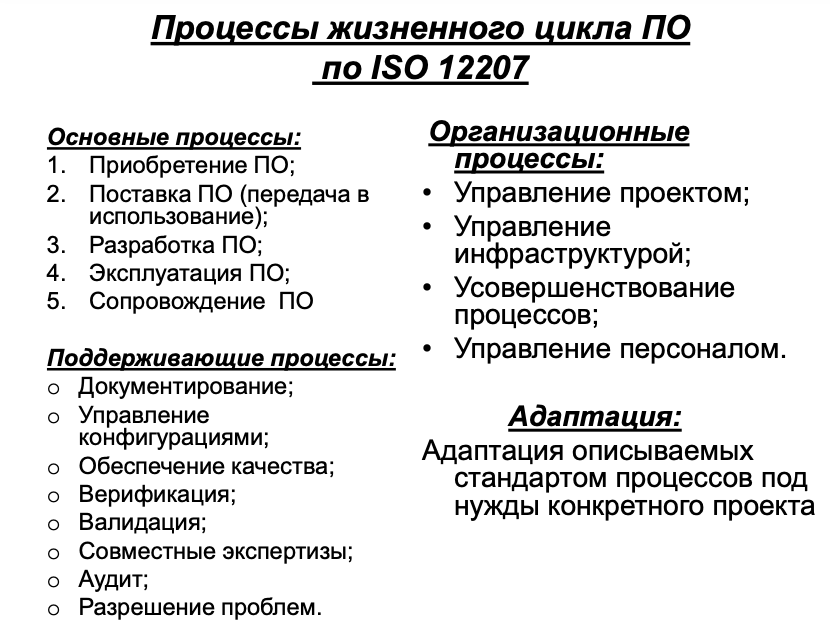
- критерии завершения соответствующей деятельности, работ и задач;

- состав отчетных материалов по этапам, затратам и графикам проведения работ;

- периодичность и способы выдачи отчетных материалов;

- состав отчетных материалов по проблемам и устраненным дефектам;

- время начала и длительность сопровождения.

****

1. **Методология обеспечения качества ПС в программной инженерии. Базовые принципы стандартов качества ПО ISO 9000:2000 и ISO 15504:1-9.**

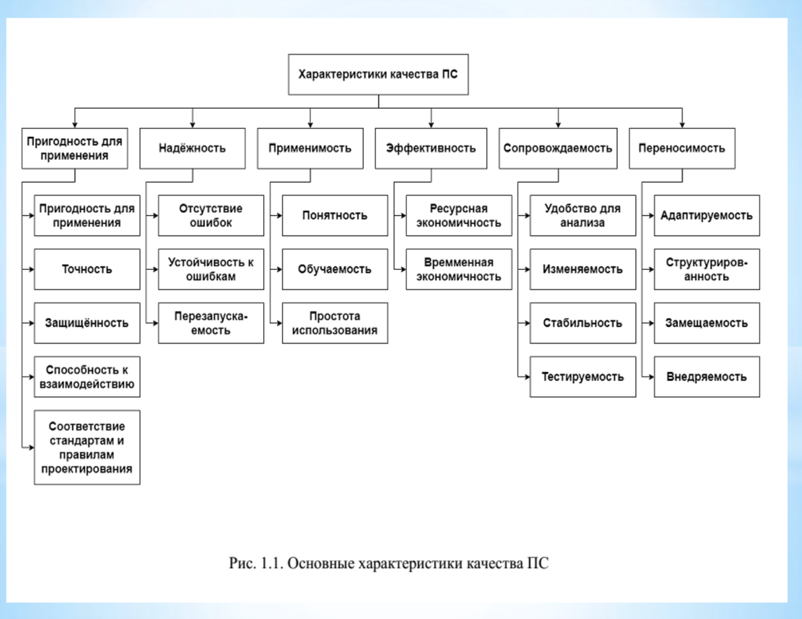
Общее представление о качестве ПС международным стандартом ISO 9126:1-4:2002 описывают 3 взаимодействующими метриками качества, отражающими:

- внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;

- внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;

- качество при использовании в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.

Модель характеристик качества ПС



Серия стандартов ISO 9000:2000 разработана, чтобы помочь предприятиям всех типов и размеров внедрить и использовать эффективные системы менеджмента (административного управления) качества.

1.ISO 9000:2000 – представляет введение в системы управления качеством продукции и услуг и словарь качества;

2.ISO 9001:2000 – устанавливает детальные требования для систем управления качеством, достаточные в случае необходимости продемонстрировать способность предприятия, обеспечить соответствие качества продукции и услуг требованиям заказчика;

3.ISO 9004:2000 – содержит руководство по внедрению и применению широко развитой системы управления качеством, чтобы достичь постоянного улучшения деловой деятельности и результатов предприятия.

Стандарт ISO 15504:1-5:2003-2006 регламентирует оценку и аттестацию зрелости процессов создания, сопровождения и совершенствования программных средств и систем, выполняемых предприятиями. Применение стандарта ISO 15504:1-5:2003-2006 направлено на выработку предприятиями и специалистами культуры постоянного совершенствования зрелости технологий обеспечения ЖЦ ПС, отвечающих бизнес-целям проектов и оптимизации использования доступных ресурсов.

1. **Управление конфигурацией в жизненном цикле программных средств. Планы, задачи и цели управления конфигурацией ПС. Отчёты о состоянии конфигурации.**

**управление конфигурацией** -- обеспечение управляемого и контролируемого развития структуры ПС, состава компонентов и функций, а также сокращение дефектов в течение всего жизненного цикла ПС.

**Процессы управления конфигурацией ПС направлены на достижения следующих основных целей:**

- обеспечить возможность оценки соответствия требованиям заказчика результатов жизненного цикла программного средства;

- обеспечить управляемую конфигурацию ПС на протяжении всего ЖЦ ;

- обеспечить управление входными и выходными данными процесса в течение всего ЖЦ, что гарантирует непротиворечивость и повторяемость работ в процессах;

- обеспечить контрольные точки для проверки, оценки состояния и контроля изменений посредством управления элементами конфигурации и определения базовой версии программного продукта;

- обеспечить контроль над тем, чтобы фиксировались дефекты и ошибки, а изменения регистрировались, утверждались и реализовались;

- гарантировать надежное физическое архивирование, восстановление и сопровождение единиц конфигурации ПС.

**Управление конфигураций включает следующие задачи:**

* подготовку процесса;
* определение конфигурации;
* контроль конфигурации;
* учет состояний конфигурации;
* оценку конфигурации;
* управление выпуском и поставку программного продукта.

**План изменений конфигурации должен содержать следующие разделы:**

- почему и с какой целью производится корректировка программ или данных;

- кто выполняет, и кто санкционирует проведение изменений комплекса программ или компонентов;

- какие действия и процедуры должны быть выполнены для реализации изменений единиц конфигурации;

- когда по срокам и в координации, с какими другими процедурами следует реализовать определенную модификацию компонентов и конфигурацию ПС;

- как и с использованием, каких средств и ресурсов должны быть выполнены запланированные изменения ПС и компонентов.

**В общем случае, отчет о конфигурации может включать следующие элементы:**

- Идентификатор продукта

- Версии продукта:

- Изменения: Список всех изменений, внесенных в продукт, включая описание изменений, даты и авторов изменений.

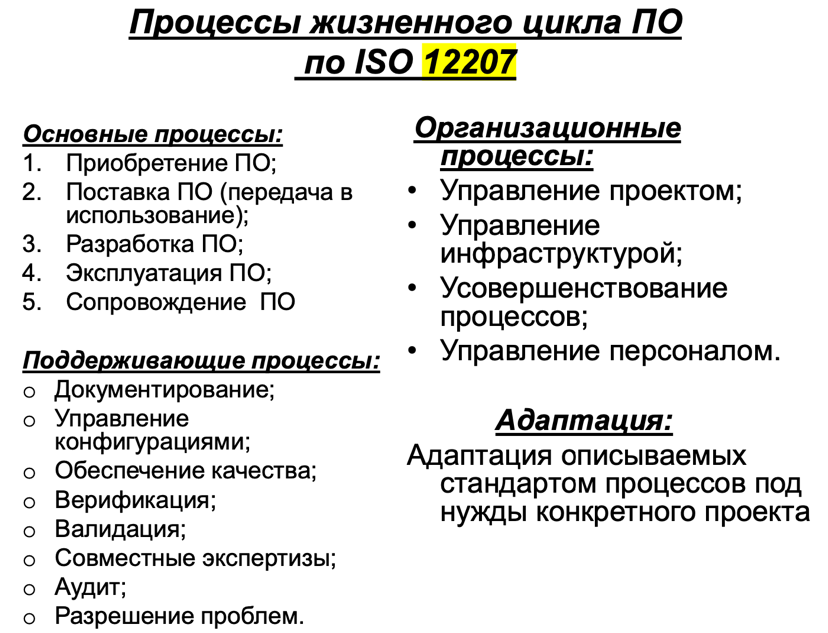
- Конфигурационные элементы: Полный список всех компонентов или элементов, включенных в конфигурацию продукта.

- Статус конфигурации: Информация о текущем состоянии конфигурации, включая любые открытые проблемы или задачи.

1. **Профили стандартов жизненного цикла систем и программных средств в программной инженерии. Профиль жизненного цикла ПС и БД на основе стандарта ISO 12207.**

Жизненным циклом ПО называют весь период существования ПО, связанный с подготовкой к его разработке, разработкой, использованием и переработками, начиная с того момента, когда принимается решение о его разработке до того момента, когда полностью прекращается всякое ее использование.

ISO/IEC 12207 (также известный как IEEE 12207) - это международный стандарт, который определяет процессы жизненного цикла программных систем. Его основное назначение заключается в том, чтобы обеспечить универсальный набор процессов для разработки и поддержки программного обеспечения. Он предоставляет общую основу для определения, управления и выполнения процессов жизненного цикла программных систем.

****

1. **Базовые характеристики и субхарактеристики качества программных средств, описанные в стандарте ISO 9126:1-4:2002.**

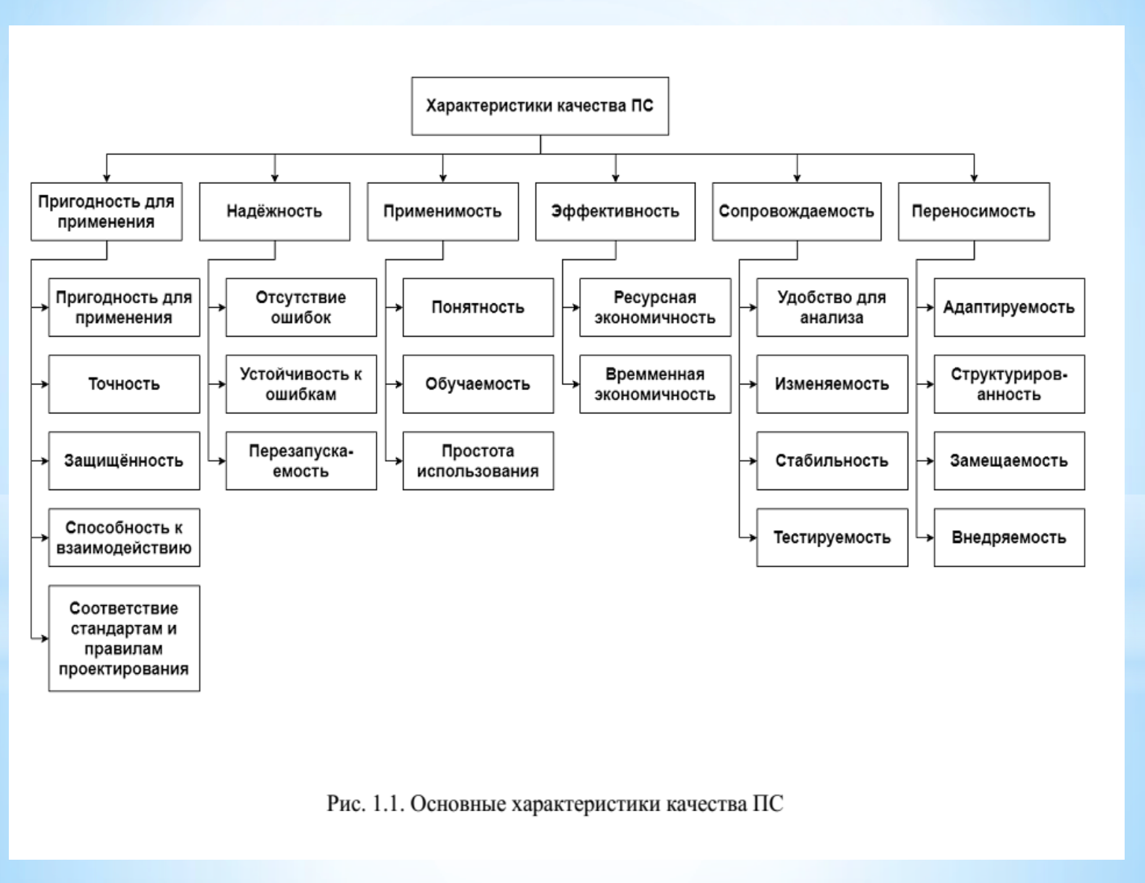
Общее представление о качестве ПС международным стандартом ISO 9126:1-4:2002 описывают 3 взаимодействующими метриками качества, отражающими:

- внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;

- внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;

- качество при использовании в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.

Модель характеристик качества ПС и компонентов состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками:



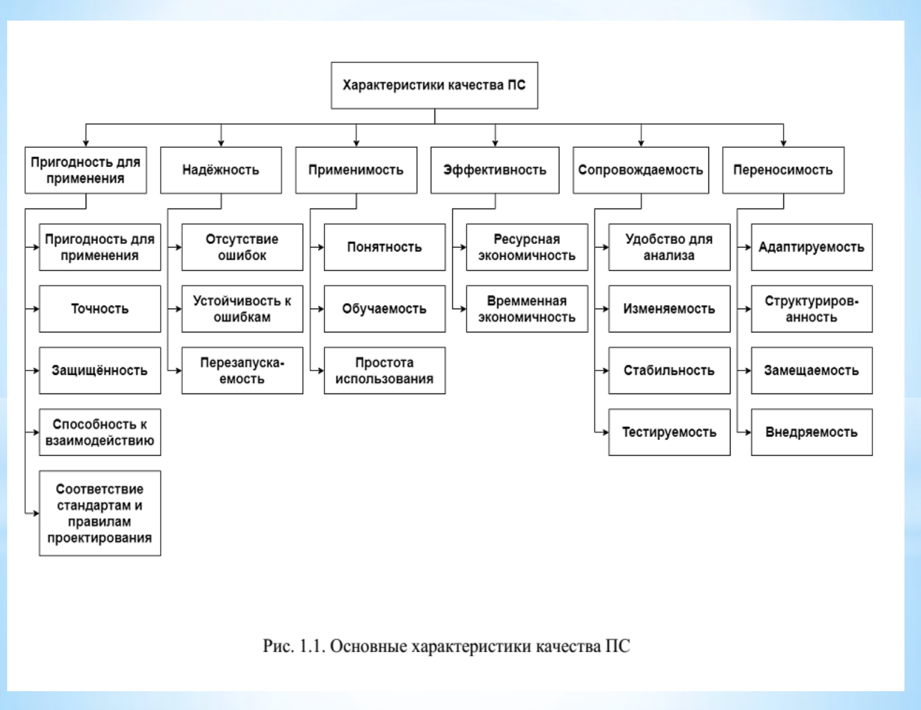
1. **Процессы сертификации в жизненном цикле программных средств. Документирование процессов и результатов сертификации программных продуктов в соответствии с рекомендациями стандартов ISO 12207, ISO 15504, ISO 9126.**

Основной целью сертификации программных средств и систем качества, обеспечивающих их жизненный цикл, является контроль и удостоверение качества технологий и продукции, гарантирование их высоких потребительских свойств.

- ISO 12207: Этот стандарт определяет процессы жизненного цикла программных систем. При документировании процессов и результатов сертификации в соответствии с ISO 12207, организации должны учитывать все этапы жизненного цикла продукта, включая планирование, разработку, тестирование, поставку, поддержку и управление конфигурацией. Документация должна содержать информацию о выполненных процессах, результатах проверок и аудитов, а также планах по управлению изменениями.

- ISO 15504 (также известный как SPICE): Этот стандарт предоставляет модель процессов оценки и улучшения способности организации разрабатывать программное обеспечение. При документировании процессов и результатов сертификации в соответствии с ISO 15504, организации должны учитывать процессы управления, инженерии и поддержки, а также соблюдение установленных практик и процедур.

- ISO 9126: Этот стандарт определяет модель качества программного обеспечения. При документировании процессов и результатов сертификации в соответствии с ISO 9126, организации должны учитывать атрибуты качества программного обеспечения, такие как функциональность, надежность, удобство использования, эффективность, удобство сопровождения и переносимость.



Документация должна включать в себя планы тестирования, отчеты об оценке качества, результаты тестирования и анализа качества продукта, а также информацию о принятых мерах по улучшению качества.

1. **Понятие открытой системы. Стандарты POSIX.**

Открытая система -- это система, имеющая внешние взаимодействия. Такие взаимодействия могут принимать форму передачи информации, энергии или материала внутрь или за пределы границы системы, в зависимости от дисциплины, определяющей это понятие.

Стандарт POSIX (Portable Operating System Interface) представляет собой набор стандартов, разработанных для обеспечения совместимости между различными операционными системами. Он определяет интерфейсы между операционной системой и прикладным программным обеспечением, что позволяет приложениям быть переносимыми между различными UNIX-подобными операционными системами.

POSIX включает в себя стандарты для системных вызовов, библиотек функций, интерфейсов командной строки, утилит и других элементов операционных систем. Он обеспечивает единый набор функций, который программисты могут использовать для написания переносимого кода, который будет работать на различных UNIX-подобных системах, таких как Linux, macOS и других.

Таким образом, стандарт POSIX играет важную роль в обеспечении переносимости программного обеспечения между различными операционными системами, что упрощает разработку и поддержку программ на различных платформах.

1. **Управление проектами программных средств в системе СMMI.**

Назначение методологии СММ/CMMI – системы и модели оценки зрелости – состоит в предоставлении необходимых общих рекомендаций и инструкций предприятиям, производящим ПС, по выбору стратегии совершенствования качества процессов и продуктов, путем анализа степени их производственной зрелости и оценивания факторов, в наибольшей степени влияющих на качество ЖЦ ПС, а также посредством выделения процессов, требующих модернизации.

Модель CMMI (Capability Maturity Model Integration) также представляет пять уровней зрелости, но она расширяет и дополняет модель CMM, включая не только процессы разработки программного обеспечения, но и другие аспекты организации. Вот краткое описание каждого уровня модели CMMI:

1. Уровень 1: Начальный (Performed)

- Организация имеет базовые процессы, которые выполняются по мере необходимости. Управление процессами находится на уровне индивидуальных усилий.

2. Уровень 2: Управляемый (Managed)

- Организация фокусируется на управлении проектами и процессами. Процессы документированы, стандартизированы и повторяемы.

3. Уровень 3: Определенный (Defined)

- Организация имеет установленные стандарты и процессы, которые описаны и управляются. Процессы разработки интегрированы с процессами управления организацией.

4. Уровень 4: Квантитативно управляемый (Quantitatively Managed)

- Организация использует количественные методы управления процессами для достижения улучшения качества продукта и процессов разработки.

5. Уровень 5: Оптимизируемый (Optimizing)

- Организация стремится к постоянному улучшению процессов через инновации, внедрение передовых технологий и автоматизацию.

1. **Оценивание надежности и безопасности функционирования сложных программных средств. Оценивание эффективности использования ресурсов ЭВМ.**

Надежность программного обеспечения - это способность системы выполнять свои функции в заданных условиях на протяжении определенного времени.

* Для оценки надежности программного обеспечения используются различные методы, такие как тестирование, анализ кода, моделирование и др.
* Оценка надежности программного обеспечения позволяет выявить потенциальные проблемы и риски, связанные с его использованием.
* Безопасность программного обеспечения:

Безопасность программного обеспечения - это защита от несанкционированного доступа, использования и изменения данных, а также от других угроз, связанных с безопасностью.

* Для обеспечения безопасности программного обеспечения применяются различные меры, такие как шифрование данных, аутентификация пользователей, контроль доступа и др.
* Оценка безопасности программного обеспечения помогает определить уязвимости и риски, которые могут быть использованы злоумышленниками для атаки на систему.

Эффективность использования ресурсов ЭВМ — это показатель, который отражает, насколько эффективно система использует доступные ресурсы, такие как процессорное время, память, дисковое пространство и т. д.

Для оценки эффективности использования ресурсов ЭВМ используются следующие методы:

* Мониторинг: позволяет отслеживать использование ресурсов в реальном времени и выявлять проблемы, связанные с их неэффективным использованием. Мониторингу подлежат такие параметры, как загрузка процессора, использование памяти, скорость чтения/записи на диск и др.
* Профилирование: метод анализа, позволяющий определить, какие части программы потребляют больше всего ресурсов. С помощью профилирования можно выявить узкие места в системе и оптимизировать её работу.

1. **Стандарт ISO 14252:1996 − Руководство по POSIX окружению открытых систем. Понятие открытой системы. Концепция стандартов POSIX, её цели и задачи.**

Стандарт ISO 14252:1996:

* В нем изложена идеология и модель создания мобильных ПС, которые детализирует для пользователей модель комплекса стандартов POSIX.
* Считается, что прикладные программы непосредственно не взаимодействуют с внешним окружением, а связаны с ним только через операционную систему.
* Модель отражает принципы построения интерфейсов прикладных программ с платформой операционной системой, через которую осуществляется взаимодействие с компонентами внешнего окружения.

Открытая система -- это система, имеющая внешние взаимодействия. Такие взаимодействия могут принимать форму передачи информации, энергии или материала внутрь или за пределы границы системы, в зависимости от дисциплины, определяющей это понятие.

Стандарт POSIX (Portable Operating System Interface) представляет собой набор стандартов, разработанных для обеспечения совместимости между различными операционными системами. Он определяет интерфейсы между операционной системой и прикладным программным обеспечением, что позволяет приложениям быть переносимыми между различными UNIX-подобными операционными системами.

POSIX включает в себя стандарты для системных вызовов, библиотек функций, интерфейсов командной строки, утилит и других элементов операционных систем. Он обеспечивает единый набор функций, который программисты могут использовать для написания переносимого кода, который будет работать на различных UNIX-подобных системах, таких как Linux, macOS и других.

Таким образом, стандарт POSIX играет важную роль в обеспечении переносимости программного обеспечения между различными операционными системами, что упрощает разработку и поддержку программ на различных платформах.

1. **Основные факторы, определяющие качество сложных программных средств. Метрики характеристик качества,описанные в стандарте ISO 9126:1-4:2002.**

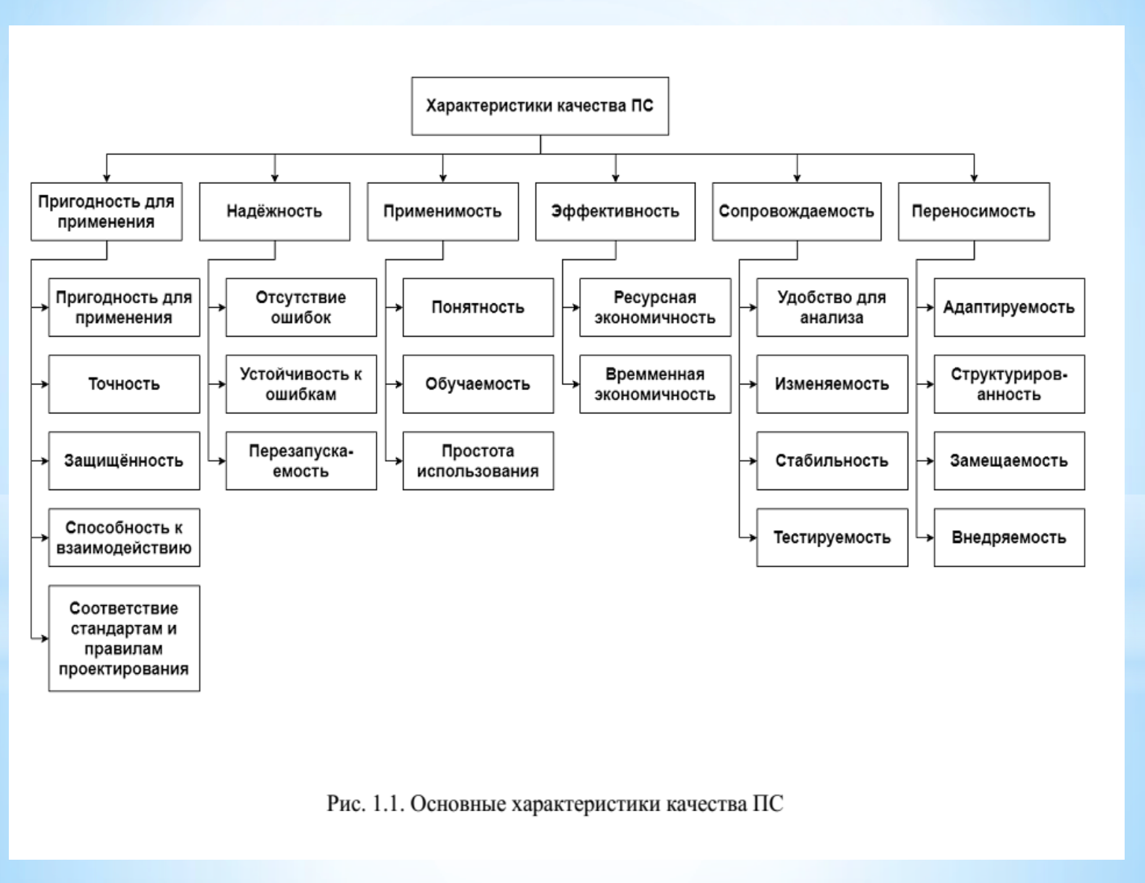
Общее представление о качестве ПС международным стандартом ISO 9126:1-4:2002 описывают 3 взаимодействующими метриками качества, отражающими:

- внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;

- внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;

- качество при использовании в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.

Модель характеристик качества ПС и компонентов состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками:



1. **Понятие сложной системы. Цели и принципы системного проектирования сложных программных средств (ПС). Процессы системного проектирования ПС.**

*1. Система, которая разрабатывается не одним человеком, а группой разработчиков (более 5 человек).*

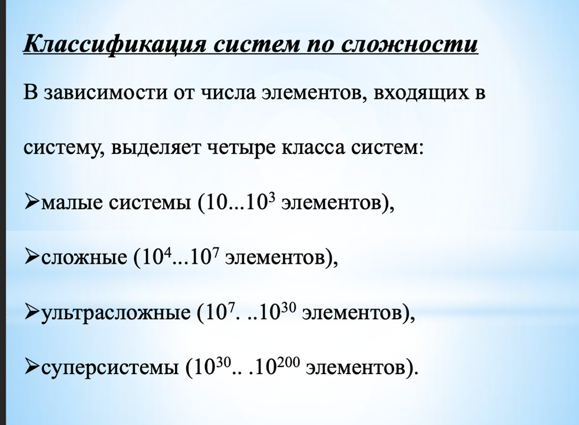
*2.Если количество строк исходного кода исчисляется сотнями тысяч или даже миллионами.*

*3.Если сложную задачу можно декомпозировать на более простые задачи (которые будут реализованы более мелкими подсистемами).*

*4.Если в требованиях встречаются взаимоисключающие требования. Например, нужно обработать огромные информационные потоки, и время отклика должно быть минимальным. Следует найти компромисс.*

* Она решает одну или несколько связанных задач,
* Она должна быть удобной в использовании.
* Должна включать достаточно полную и понятную пользователям документацию
* Должна включать набор документов для обучения пользователей работе с программой.
* Ее низкая производительность на реальных данных приводит к значимым потерям для пользователей.
* Ее неправильная работа наносит ощутимый ущерб пользователям и другим организациям и лицам, даже если сбои происходят не слишком часто.
* Для выполнения своих задач она должна взаимодействовать с другими программами и программно-аппаратными системами, работать на разных платформах

Велико количество ее возможных пользователей.



Цели системного проектирования ПС:

1. Создание эффективной и надежной системы: Целью системного проектирования является разработка программных средств, которые обеспечивают высокую производительность, надежность и безопасность при выполнении задач.

2. Удовлетворение потребностей пользователей: Системное проектирование направлено на создание ПС, которые соответствуют требованиям и ожиданиям пользователей, обеспечивая удобство использования и функциональность.

3. Масштабируемость и гибкость: Целью является создание ПС, способных масштабироваться и адаптироваться к изменяющимся потребностям бизнеса или пользователей.

4. Эффективное использование ресурсов: Системное проектирование направлено на оптимизацию использования ресурсов, таких как память, процессорное время и сетевые ресурсы.

Принципы системного проектирования ПС:

1. Модульность: Разделение системы на независимые модули для упрощения разработки, тестирования и сопровождения.

2. Интеграция: Обеспечение взаимодействия между модулями для создания целостной системы.

3. Интерфейсы: Создание четких интерфейсов между компонентами системы для упрощения их взаимодействия.

Контекст:

1. Постановка задачи

2. Анализ требований: Сбор и анализ требований пользователей для определения функциональности и характеристик системы.

*3. Проектирование архитектуры: Разработка общей структуры системы, включая определение компонентов, интерфейсов и взаимодействия между ними. Определение состава необходимого оборудования, программного обеспечения и операций, выполняемых обслуживающим персоналом.*

*Архитектурой ПО называют совокупность базовых концепций (принципов)построения программного обеспечения.*

*4. Детальное проектирование: Проектирование отдельных модулей и компонентов системы с учетом технических деталей.*

5. Разработка и интеграция: Создание кода для каждого модуля и последующая интеграция этих модулей в целостную систему.

6. Сопровождение



Также в этап проектирования входит выбор парадигмы программирования.

1. **Оценивание надежности и безопасности функционирования сложных программных средств. Оценивание эффективности использования ресурсов ЭВМ.**

Надежность программного обеспечения - это способность системы выполнять свои функции в заданных условиях на протяжении определенного времени.

* Для оценки надежности программного обеспечения используются различные методы, такие как тестирование, анализ кода, моделирование и др.
* Оценка надежности программного обеспечения позволяет выявить потенциальные проблемы и риски, связанные с его использованием.
* Безопасность программного обеспечения:

Безопасность программного обеспечения - это защита от несанкционированного доступа, использования и изменения данных, а также от других угроз, связанных с безопасностью.

* Для обеспечения безопасности программного обеспечения применяются различные меры, такие как шифрование данных, аутентификация пользователей, контроль доступа и др.
* Оценка безопасности программного обеспечения помогает определить уязвимости и риски, которые могут быть использованы злоумышленниками для атаки на систему.

Эффективность использования ресурсов ЭВМ — это показатель, который отражает, насколько эффективно система использует доступные ресурсы, такие как процессорное время, память, дисковое пространство и т. д.

Для оценки эффективности использования ресурсов ЭВМ используются следующие методы:

* Мониторинг: позволяет отслеживать использование ресурсов в реальном времени и выявлять проблемы, связанные с их неэффективным использованием. Мониторингу подлежат такие параметры, как загрузка процессора, использование памяти, скорость чтения/записи на диск и др.
* Профилирование: метод анализа, позволяющий определить, какие части программы потребляют больше всего ресурсов. С помощью профилирования можно выявить узкие места в системе и оптимизировать её работу.

1. **Организация документирования программных средств. Типы документаций.**

Процессы документирования программ и данных входят в весь ЖЦ сложных систем и ПС. Поэтому организация и реализация работ по созданию документов должны распределяться между специалистами, ведущими непосредственное и преимущественное создание проектов комплексов.

При создании особо сложных систем целесообразно выделение специального коллектива, обеспечивающего организацию и реализацию основных системных работ по документообороту ПС.

Совокупные затраты на документирование крупных программных продуктов могут достигать 20 – 30% от общей трудоемкости проекта и необходимого числа (десятки) специалистов в жизненном цикле проекта ПС.

По своему назначению и ориентации на определенные задачи и группы пользователей, документацию ПС можно разделить:

* Технологическую документацию процессов разработки и обеспечения всего ЖЦ, включающую подробные технические описания.

Обеспечивает возможность отчуждения, детального освоения, развития и корректировки ими программ и данных на всем жизненном цикле ПС;

* Эксплуатационную документацию объекта и результатов разработки, создаваемую для конечных пользователей ПС и позволяющую им осваивать и квалифицированно применять эти средства для решения конкретных функциональных задач систем.

Эксплуатационная документация должна обеспечивать отчуждаемость программного продукта от первичных разработчиков и возможность освоения и эффективного применения комплексов программ достаточно квалифицированными специалистами – пользователями ПС и системы.

1. **Задачи, решаемые с помощью машинного обучения и искусственного интеллекта. Обоснование необходимости использования *объяснимого искусственного интеллекта (XAI*). Существующие решения для XAI. Проблемы, связанные с развитием и использованием XAI.**

Все задачи, решаемые с помощью машинного обучения (machine learning, ML) и искусственного интеллекта (artificial intelligence, AI), относятся к одной из следующих категорий:

1) Задача регрессии – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число.

2) Задача классификации – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов (как правило, в формате «да» или «нет»).

3) Задача кластеризации – распределение данных на группы.

4) Задача уменьшения размерности – сведение большого числа признаков к меньшему (обычно 2–3) для удобства их последующей визуализации.

5) Задача выявления аномалий – выявление отклонений от стандартных случаев. Эта задача похожа на задачу классификации, но более сложна в обучении.

Модели искусственного интеллекта/машинного обучения – это математические алгоритмы, которые с помощью вклада человека (эксперта) «обучаются» на наборах данных воспроизводить решение, которое эксперт примет при анализе такого же набора данных. Обучившись воспроизводить решение эксперта, модель в дальнейшем функционирует самостоятельно, позволяя таким образом автоматизировать решение задач. В идеале модель должна также обосновывать своё решение, чтобы помочь интерпретировать процесс принятия решения.

Объяснимый искусственный интеллект (Explainable AI, XAI) – модель, которая могла бы в перспективе объяснять механизмы, лежащие за алгоритмами машинного обучения.

Множество решений, применяющих алгоритмы ИИ, представляют собой подобие «черного ящика», – зачастую не только конечные пользователи, но и сами разработчики не могут точно определить, как именно модель машинного обучения пришла к тем или иным выводам в ходе обработки исходных данных.

Объяснимость ИИ будет включать в себя три составляющие: симулируемость, разложимость, алгоритмическую прозрачность.

- Симулируемость означает возможность анализа модели человеком. Наиболее важным критерием для симулируемости является сложность модели. Простые, но обширные (со слишком большим количеством правил) системы, основанные на правилах, не соответствуют этой характеристике, тогда как одиночная нейронная сеть персептрона попадает в нее.

- Разложимость означает способность объяснить каждую из частей модели (входные данные, параметры и выходные данные). Громоздкие функции не соответствуют данному критерию.

- Алгоритмическая прозрачность означает способность пользователя понять процесс, которому следует модель ИИ, чтобы произвести любой заданный вывод из ее входных данных.

Существует несколько различных подходов для решения проблемы объяснимости ИИ.

1. Локальные объяснения сегментируют пространство решений и дают объяснения менее сложным подпространствам решений, которые актуальны для всей модели. Эти объяснения могут быть сформированы с помощью методов, которые объясняют часть функционирования всей системы.

2. Объяснения на примерах предполагают извлечение репрезентативных примеров, которые улавливают внутренние отношения и корреляции, обнаруживаемые анализируемой моделью данных, и относятся к результату, сгенерированному определенной моделью.

3. Объяснения посредством упрощения в совокупности обозначают те методы, в которых целая новая система ИИ перестраивается на основе обученной модели ИИ, которую необходимо объяснить. Новый ИИ обычно пытается оптимизировать свое сходство с изначальной моделью ИИ, уменьшая при этом ее сложность, но сохраняя тот же уровень производительности.

4. Методы апостериорного объяснения релевантности признаков проясняют внутреннее функционирование модели, вычисляя оценку релевантности для ее управляемых переменных. Эти оценки количественно определяют влияние (чувствительность) на выходные данные модели. Сравнение оценок различных переменных показывает вес, которую модель присваивает каждой из таких переменных при получении результатов.

5. Отсутствие объяснимости может приводить к ситуациям, когда модель присваивает более высокий вес тем входным переменным, которые объективно не должны иметь такого веса.

Примеры:

Злоумышленники могут добавлять небольшие искажения в изображения, которые не повлияют на решение человека, но могут сбить с толку алгоритм, ранее доказавший свою эффективность на не измененных искусственно изображениях (рис. 11).

Этические проблемы, связанные с работой моделей черного ящика, возникают из-за их склонности непреднамеренно принимать несправедливые решения с учетом чувствительных факторов, таких как раса, возраст или пол человека.Пример: система оценки риска повторного совершения преступных деяний COMPAS принимала расу как одну из важных характеристик для принятия решения о высоком риске рецидива.

Объяснения алгоритмов работы ИИ могут быть представлены в текстовой или визуальной форме.

- Текстовые объяснения представляют собой метод создания символов, отображающих логику алгоритма посредством семантического отображения.

- Многие из методов визуализации сопровождаются методами уменьшения размерности для упрощения понимания работы модели человеком. Методы визуализации могут сочетаться с другими методами для улучшения их понимания и считаются наиболее подходящим способом представить сложные взаимодействия между переменными, участвующими в модели.

Проблемы, связанные с развитием и использованием XAI

- создание более понятной модели машинного обучения может в конечном итоге ухудшить качество принимаемых ею решений.

- Более сложные модели обладают гораздо большей гибкостью, чем их более простые аналоги, что позволяет аппроксимировать более сложные функции.

- Дилемма аппроксимации тесно связана с интерпретируемостью: объяснения, сделанные для модели машинного обучения, должны соответствовать требованиям аудитории, для которой они формируются, обеспечивая репрезентативность исследуемой модели без излишнего упрощения ее основных черт.

В науке в отношении XAI еще не существует единых объективных показателей того, что конкретно является хорошим объяснением.

1. **Верификация, тестирование и оценивание корректности программных компонентов. Цели, задачи и назначение. Виды и методы тестирования.**

Верификация -- это процесс для определения, выполняют ли программные средства и их компоненты требования, наложенные на них в последовательных этапах ЖЦ ПС.

Основная цель верификации ПС состоит в том, чтобы обнаружить, зарегистрировать и устранить дефекты и ошибки, которые внесены во время последовательной разработки или модификации программ.

Тестирование ПС от требований имеет две взаимодополняющие цели:

* Показать, что ПС удовлетворяет заданным требованиям к нему.
* Показать с высокой степенью доверия, что устранены дефекты и ошибки, которые могли бы привести к возникновению отказов, влияющих на корректность и надежность системы.

При первой стратегии тестирования за основу принимается структура ПМ, построенная по тексту программы в виде графа (УГП).

1. В УПГ выделяются и упорядочиваются маршруты исполнения программ.
2. Эти условия используются для подготовки тестовых наборов, каждый из которых должен реализоваться по маршруту, принятому за эталон при подготовке теста.
3. Отклонение исполнения теста от первоначально выбранного маршрута рассматривается как ошибка

При второй стратегии за основу принимаются требования спецификаций, конкретные тестовые и эталонные значения.

1. При каждом тесте программа исполняется по определенному маршруту, который регистрируется.
2. По мере тестирования отмечаются проверенные операторы, и оценивается полнота покрытия тестами требований спецификаций на маршрутах тестирования

Программы в процессе тестирования используются в двух различных

формах представления:

1. текст программы на языке программирования или формализованного описания спецификаций требований (графическое представление);

2. Текст в машинном коде конкретной ЭВМ (объектное представление).

Системы автоматизации тестирования можно разделить на два вида:

* статические системы, которые анализируют спецификации и исходные тексты программ в объектном коде без их исполнения на ЭВМ (анализаторы);
* Динамические системы, при использовании которых программы функционируют в объектном коде и пригодны для их реального применения на ЭВМ.

В группу средств статической отладки входят также средства расчета длительностей исполнения модулей и компонентов. В результате расчетов выявляются компоненты программы, требующие избыточно большого времени счета на реализующей ЭВМ.

Группу средств динамической отладки можно разделить на два типа:

1. Основные средства, непосредственно обеспечивающие исполнение программ в соответствии с отладочными заданиями (транслятор, исполнение по сценарию, регистрация результатов).

2. Вспомогательные средства, которые анализируют выполненное тестирование, его результаты и проведенные корректировки.

1. **Национальная стратегия развития искусственного интеллекта и нейротехнологий на период до 2030 года. Стандарты по защите интеллектуальной собственности в области программной инженерии.**

Основные задачи стратегии:

* поддержка научных исследований в области ИИ;
* разработка и развитие программного обеспечения, в котором используются технологии ИИ;
* повышение доступности и качества данных;
* повышение доступности аппаратного обеспечения;
* повышение уровня обеспечения российского рынка технологий ИИ квалифицированными кадрами;
* повышение уровня информированности населения о возможных сферах использования технологий ИИ;
* создание системы регулирования общественных отношений, возникающих в связи с развитием и использованием технологий ИИ.

Для реализации стратегии необходимо:

разработка единых стандартов в области безопасности и совместимости программного обеспечения;

создание нормативно-правовой базы, предусматривающей обеспечение защиты данных, полученных при осуществлении экономической и научной деятельности.

Стандарты по защите интеллектуальной собственности в области программной инженерии направлены на обеспечение правовой защиты результатов интеллектуального труда разработчиков программного обеспечения. Эти стандарты помогают предотвратить несанкционированное использование, копирование и распространение программного продукта, а также защищают авторские права разработчиков.

К основным стандартам по защите интеллектуальной собственности относятся:

ISO/IEC 20071:2019 «Информационные технологии — Методы обеспечения безопасности — Системы управления информационной безопасностью» — стандарт, который определяет требования к системам управления информационной безопасностью для организаций, занимающихся разработкой программного обеспечения. Стандарт помогает обеспечить защиту интеллектуальной собственности от несанкционированного доступа, изменения или уничтожения.

ГОСТ Р 56829-2015 «Защита информации. Автоматизированные системы в защищённом исполнении. Управление доступом к объектам» — российский стандарт, определяющий требования к управлению доступом к объектам автоматизированных систем в защищённом исполнении, включая программное обеспечение. Стандарт обеспечивает защиту от несанкционированного использования, копирования и распространения программного продукта.

NIST SP 800-53 «Рекомендации по управлению рисками для федеральных информационных систем и организаций» — американский стандарт, который содержит рекомендации по управлению рисками, связанными с защитой информации, включая интеллектуальную собственность. Стандарт может быть использован при разработке и внедрении систем управления информационной безопасностью в организациях, занимающихся программной инженерией.

OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data — рекомендации Организации экономического сотрудничества и развития (ОЭСР) по защите персональных данных, которые могут включать информацию об интеллектуальной собственности разработчиков программного обеспечения. Рекомендации могут быть использованы при разработке политики и процедур по защите конфиденциальной информации в организациях, занимающихся разработкой ПО.

Эти стандарты обеспечивают правовую основу для защиты интеллектуальной собственности и способствуют созданию безопасной и надёжной среды для разработки и использования программного обеспечения. Они помогают разработчикам и организациям защитить свои авторские права, предотвратить нарушение прав интеллектуальной собственности других лиц и обеспечить соблюдение законодательства в области защиты авторских прав.

1. **Цели и процессы технико-экономического обоснования проектов программных средств. Методика 1 – экспертное технико-экономическое обоснование проектов программных средств.**

Технико-экономическое обоснование (ТЭО) — это документ, который содержит анализ экономической целесообразности реализации проекта программных средств. ТЭО помогает определить, стоит ли вкладывать ресурсы в разработку и внедрение программного продукта, а также оценить его потенциальную прибыльность и риски.

Цели технико-экономического обоснования проектов программных средств:

Определить экономическую целесообразность разработки и внедрения программного продукта.

Оценить затраты на разработку, внедрение и поддержку программного продукта.

Рассчитать ожидаемую прибыль от реализации программного продукта.

Проанализировать риски, связанные с реализацией программного продукта.

Для достижения этих целей необходимо провести детальный анализ всех аспектов проекта, включая:

Описание проекта: цели, задачи, ожидаемые результаты.

Анализ рынка: спрос на программный продукт, конкуренция, потенциальные клиенты.

Оценка затрат: разработка, тестирование, внедрение, поддержка.

Расчёт прибыли: доходы от продажи лицензий, услуг поддержки, дополнительных функций.

Риски: технические, финансовые, юридические.

Методика 1 — экспертное технико-экономическое обоснование проектов программных средств — предполагает использование опыта и знаний экспертов для оценки экономической целесообразности проекта. Эксперты могут быть привлечены из различных областей, таких как программирование, бизнес-анализ, маркетинг, финансы и т. д.

Процесс экспертного технико-экономического обоснования включает следующие шаги:

Сбор информации о проекте: описание проекта, цели, задачи, ожидаемые результаты, требования к функциональности, сроки реализации.

Анализ рынка: исследование спроса на программные продукты подобного типа, изучение конкурентов, определение потенциальных клиентов.

Оценка затрат: расчёт затрат на разработку, тестирование, внедрение и поддержку проекта. Затраты могут включать в себя оплату труда разработчиков, аренду оборудования, покупку лицензий на программное обеспечение, расходы на обучение персонала и т. п.

Расчёт прибыли: оценка доходов от продажи лицензий на программный продукт, услуг технической поддержки, дополнительных функций и возможностей.

Риски: выявление и оценка технических, финансовых и юридических рисков, связанных с проектом.

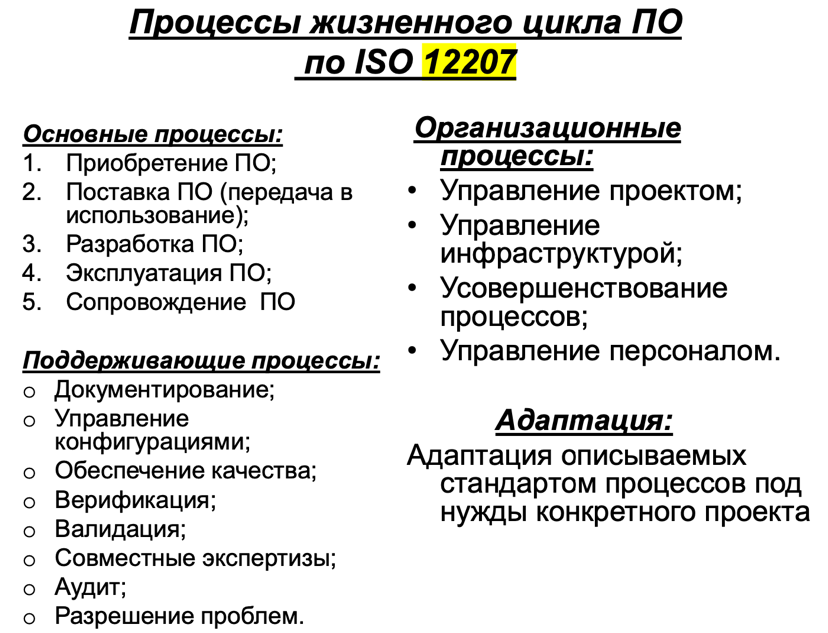
Принятие решения: на основе проведённого анализа эксперты делают вывод о целесообразности реализации проекта и дают рекомендации по его оптимизации.

Экспертное технико-экономическое обоснование является важным этапом в процессе разработки и реализации проектов программных средств. Оно позволяет оценить экономическую эффективность проекта, выявить возможные проблемы и риски, а также принять обоснованное решение о его реализации.

1. **Организация планирования жизненного цикла сложных программных средств в соответствии со стандартами ISO 12207 и ISO 16326.**

Жизненным циклом ПО называют весь период существования ПО, связанный с подготовкой к его разработке, разработкой, использованием и переработками, начиная с того момента, когда принимается решение о его разработке до того момента, когда полностью прекращается всякое ее использование.

ISO/IEC 12207 (также известный как IEEE 12207) - это международный стандарт, который определяет процессы жизненного цикла программных систем. Его основное назначение заключается в том, чтобы обеспечить универсальный набор процессов для разработки и поддержки программного обеспечения. Он предоставляет общую основу для определения, управления и выполнения процессов жизненного цикла программных систем.

****

Стандарт ISO/IEC 16326 дополняет ISO/IEC 12207, предоставляя модель процессов жизненного цикла программных средств и предписывая методы их применения.

1. Моделирование процессов: Использование моделей для описания и анализа процессов жизненного цикла программных средств.

2. Управление рисками: Оценка рисков и разработка стратегий управления рисками на всех этапах жизненного цикла ПС.

3. Конфигурационное управление: Установление и поддержание системы управления конфигурациями для контроля версий и изменений в ПС.

4. Управление требованиями: Определение, анализ, утверждение и управление требованиями к ПС.

5. Управление качеством: Планирование, обеспечение и контроль качества программных средств на всех этапах жизненного цикла.

Оба стандарта предоставляют подробные рекомендации по организации планирования жизненного цикла программных средств, обеспечивая стандартизированный подход к разработке ПС с учетом требований качества, управления рисками и удовлетворения потребностей пользователей.

1. **Организация разработки требований к сложным программным средствам. Процессы разработки требований к характеристикам сложных программных систем.**

основные свойства требований в ТЗ:

1. Требование должно быть понятным.

2. Требование должно быть конкретным.

3. Требование должно быть тестируемым.

Если результат выполнения требования невозможно протестировать, значит, оно либо не понятное, либо не конкретное.

Техническое задание – это документ, в основе которого лежат требования, сформулированные на понятном (привычном) для Заказчика языке.

При этом должна использоваться отраслевая терминология, понятная Заказчику.

Никаких привязок к особенностям технической реализации быть не должно.

 На этапе ТЗ не важно, на какой платформе будут реализовываться эти требования.

Однако, если речь идет о внедрении системы на основе уже существующего ПО, то такая привязка может иметь место, но только на уровне экранных форм, форм отчетов и прочего. Выяснением и формулированием требований, а также разработкой Технического задания должен заниматься бизнес- аналитик, а также возможно участие программиста.

ГОСТ рекомендует следующие разделы ТЗ:

- общие сведения;

- назначение и цели создания (развития) системы;

- характеристика объектов автоматизации -- краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию.;

- требования к системе:

* требования к структуре и функционированию системы;
* требования к численности и квалификации персонала системы
* режиму работы системы и персонала;
* характеристики качества системы;
* требования к надежности;
* требования безопасности;
* требования к эргономике и технической эстетике;
* требования к транспортабельности для подвижных АС;
* требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
* требования к защите информации от несанкционированного доступа;
* требования по сохранности информации при авариях;
* требования к защите от влияния внешних воздействий;
* требования к патентной чистоте;
* требования по стандартизации и унификации;
* к квалификации;

- состав и содержание работ по созданию системы;

- порядок контроля и приемки системы;

- требования к составу и содержанию работ по

подготовке объекта автоматизации к вводу

системы в действие;

- требования к документированию;

- источники разработки.

1. **Верификация, тестирование и оценивание корректности программных компонентов. Цели, задачи и назначение. Виды и методы тестирования.**

Верификация -- это процесс для определения, выполняют ли программные средства и их компоненты требования, наложенные на них в последовательных этапах ЖЦ ПС.

Основная цель верификации ПС состоит в том, чтобы обнаружить, зарегистрировать и устранить дефекты и ошибки, которые внесены во время последовательной разработки или модификации программ.

Тестирование ПС от требований имеет две взаимодополняющие цели:

* Показать, что ПС удовлетворяет заданным требованиям к нему.
* Показать с высокой степенью доверия, что устранены дефекты и ошибки, которые могли бы привести к возникновению отказов, влияющих на корректность и надежность системы.

При первой стратегии тестирования за основу принимается структура ПМ, построенная по тексту программы в виде графа (УГП).

1. В УПГ выделяются и упорядочиваются маршруты исполнения программ.
2. Эти условия используются для подготовки тестовых наборов, каждый из которых должен реализоваться по маршруту, принятому за эталон при подготовке теста.
3. Отклонение исполнения теста от первоначально выбранного маршрута рассматривается как ошибка

При второй стратегии за основу принимаются требования спецификаций, конкретные тестовые и эталонные значения.

1. При каждом тесте программа исполняется по определенному маршруту, который регистрируется.
2. По мере тестирования отмечаются проверенные операторы, и оценивается полнота покрытия тестами требований спецификаций на маршрутах тестирования

Программы в процессе тестирования используются в двух различных

формах представления:

1. текст программы на языке программирования или формализованного описания спецификаций требований (графическое представление);

2. Текст в машинном коде конкретной ЭВМ (объектное представление).

Системы автоматизации тестирования можно разделить на два вида:

* статические системы, которые анализируют спецификации и исходные тексты программ в объектном коде без их исполнения на ЭВМ (анализаторы);
* Динамические системы, при использовании которых программы функционируют в объектном коде и пригодны для их реального применения на ЭВМ.

В группу средств статической отладки входят также средства расчета длительностей исполнения модулей и компонентов. В результате расчетов выявляются компоненты программы, требующие избыточно большого времени счета на реализующей ЭВМ.

Группу средств динамической отладки можно разделить на два типа:

1. Основные средства, непосредственно обеспечивающие исполнение программ в соответствии с отладочными заданиями (транслятор, исполнение по сценарию, регистрация результатов).

2. Вспомогательные средства, которые анализируют выполненное тестирование, его результаты и проведенные корректировки.

1. **Основные факторы, определяющие качество сложных программных средств. Метрики характеристик качества,описанные в стандарте ISO 9126:1-4:2002.**

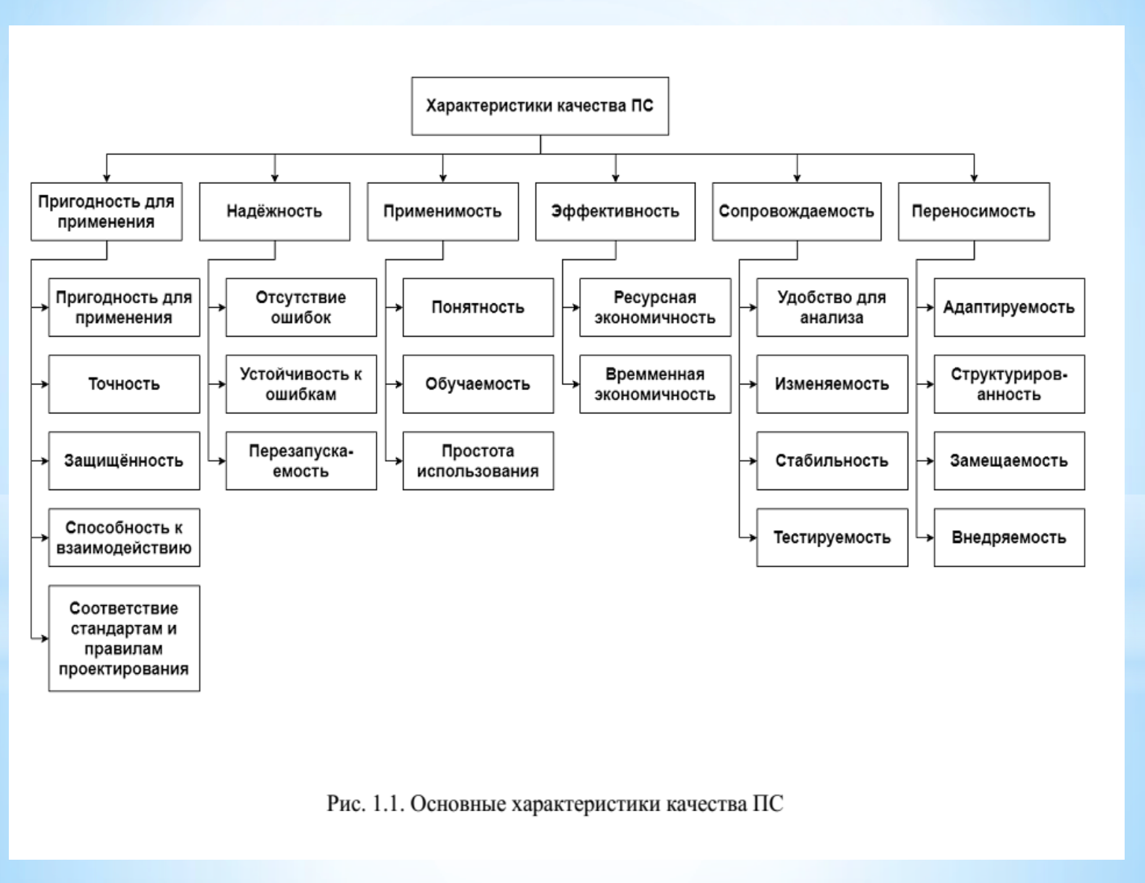
Общее представление о качестве ПС международным стандартом ISO 9126:1-4:2002 описывают 3 взаимодействующими метриками качества, отражающими:

- внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;

- внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;

- качество при использовании в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.

Модель характеристик качества ПС и компонентов состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками:



1. **Процессы переноса программных средств и баз данных,стандартизированные в ISO 14764, который детализирует требования к процессам переноса, определенным в базовом стандарте на жизненный цикл ПС (ISO 12207).**

Процессы переноса программных средств и баз данных регламентируются рядом процедур и документов, стандартизированных в ISO 14764.

Специалисты, которые проводят перенос по рекомендациям этих стандартов, должны:

1) разработать план переноса,

2)известить пользователей,

3)обучить персонал,

4)предупредить о завершении переноса, 5)оценить влияние новой версии и внешней среды,

6) архивировать соответствующие данные.

отчетными документами по переносу ПС и БД являются:

1. Перенесенный программный продукт на новой платформе;

2. План реализации переноса;

3. Инструментальные средства для

переноса;

4. Извещения о намерениях по переносу;

5. Уведомление о завершении переноса;

6. Архивные данные процессов и

результатов переноса.

1. **Дефекты, ошибки и риски в жизненном цикле программных средств. Типы ошибок и дефектов. Риски в жизненном цикле сложных программных средств.**

Статистика ошибок и дефектов в комплексах программ и их характеристики в конкретных типах проектов ПС, могут служить ориентирами для разработчиков при распределении ресурсов в жизненном цикле ПС и предохранять их от излишнего оптимизма при оценке достигнутого качества программных продуктов.

К понятию риски относятся негативные события и их величины, отражающие потери, убытки или ущерб от процессов или продуктов, вызванные дефектами при проектировании требований, недостатками обоснования проектов ПС, а также при последующих этапах разработки, реализации и всего жизненного цикла комплексов программ.

в общем случае под ошибкой подразумевается неправильность, погрешность или неумышленное искажение объекта или процесса, что может быть причиной ущерба – риска при функционировании и применении программы.

Классификация ошибок

\*Небольшими ошибками называют такие, на которые средний пользователь может и не обратить внимания при применении ПС. \*Небольшие ошибки могут включать орфографические ошибки на экране, пропущенные разделы в справочнике и другие мелкие проблемы.

\*По десятибалльной шкале рисков небольшие ошибки находятся в пределах от 1 до 3 приоритета.

Умеренными ошибками называют те, которые влияют на конечного пользователя, но имеются слабые последствия или обходные пути, позволяющие сохранить достаточную функциональность ПС.

К умеренным ошибкам относятся дефекты:

неверные ссылки на страницах,

 ошибочный текст на экране,

сбои, если их трудно воспроизвести, и они не

оказывают влияния на существенное число

пользователей.

Ошибки, которые можно исправить на этом уровне,

следует исправлять, если на это есть время и

возможность.

По десятибалльной шкале умеренные ошибки находятся в диапазоне от 4 до 7 приоритета.

Критические ошибки останавливают выпуск версии программного продукта!

Это могут быть ошибки с высоким влиянием, которые вызывают сбой в системе или потерю данных, отражаются на надежности и безопасности применения ПС, с которыми никогда не передается комплекс программ пользователю.

По десятибалльной шкале – от 8 до 10 приоритета.

Причины возникновения и проявления рисков:

1. Злоумышленные, активные воздействия заинтересованных лиц.

Для решения этой проблемы созданы и активно развиваются методы, средства и стандарты обеспечения защиты программ и данных от предумышленных негативных внешних воздействий.

2. Риски при случайных, дестабилизирующих воздействиях дефектов ПС и отсутствии предумышленного негативного влияния на системы, ПС или информацию баз данных.

Для устранения или снижения рисков до допустимых пределов может быть необходимо изменение требований к функциональной пригодности, к конструктивным характеристикам и доступным ресурсам.

1. **Стандарт ISO 14252:1996 − Руководство по POSIX окружению открытых систем.**

Стандарт ISO 14252:1996:

* В нем изложена идеология и модель создания мобильных ПС, которые детализирует для пользователей модель комплекса стандартов POSIX.
* Считается, что прикладные программы непосредственно не взаимодействуют с внешним окружением, а связаны с ним только через операционную систему.
* Модель отражает принципы построения интерфейсов прикладных программ с платформой операционной системой, через которую осуществляется взаимодействие с компонентами внешнего окружения.

Открытая система -- это система, имеющая внешние взаимодействия. Такие взаимодействия могут принимать форму передачи информации, энергии или материала внутрь или за пределы границы системы, в зависимости от дисциплины, определяющей это понятие.

Стандарт POSIX (Portable Operating System Interface) представляет собой набор стандартов, разработанных для обеспечения совместимости между различными операционными системами. Он определяет интерфейсы между операционной системой и прикладным программным обеспечением, что позволяет приложениям быть переносимыми между различными UNIX-подобными операционными системами.

POSIX включает в себя стандарты для системных вызовов, библиотек функций, интерфейсов командной строки, утилит и других элементов операционных систем. Он обеспечивает единый набор функций, который программисты могут использовать для написания переносимого кода, который будет работать на различных UNIX-подобных системах, таких как Linux, macOS и других.

Таким образом, стандарт POSIX играет важную роль в обеспечении переносимости программного обеспечения между различными операционными системами, что упрощает разработку и поддержку программ на различных платформах.

1. **Понятие сложной системы. Цели и принципы системного проектирования сложных программных средств. Процессы системного проектирования программных средств.**

**Разработка требований к программным средствам. Создание технического задания.**

*1. Система, которая разрабатывается не одним человеком, а группой разработчиков (более 5 человек).*

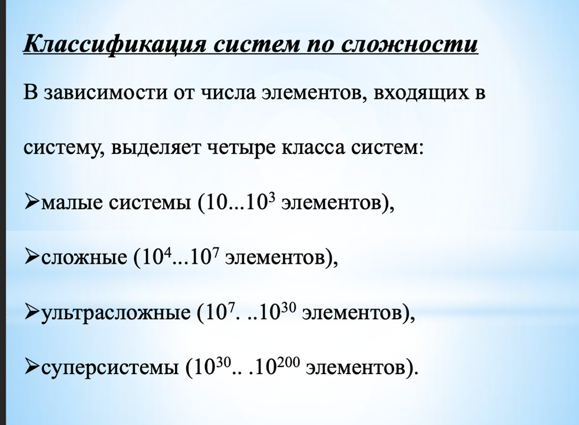
*2.Если количество строк исходного кода исчисляется сотнями тысяч или даже миллионами.*

*3.Если сложную задачу можно декомпозировать на более простые задачи (которые будут реализованы более мелкими подсистемами).*

*4.Если в требованиях встречаются взаимоисключающие требования. Например, нужно обработать огромные информационные потоки, и время отклика должно быть минимальным. Следует найти компромисс.*

* Она решает одну или несколько связанных задач,
* Она должна быть удобной в использовании.
* Должна включать достаточно полную и понятную пользователям документацию
* Должна включать набор документов для обучения пользователей работе с программой.
* Ее низкая производительность на реальных данных приводит к значимым потерям для пользователей.
* Ее неправильная работа наносит ощутимый ущерб пользователям и другим организациям и лицам, даже если сбои происходят не слишком часто.
* Для выполнения своих задач она должна взаимодействовать с другими программами и программно-аппаратными системами, работать на разных платформах

Велико количество ее возможных пользователей.



Цели системного проектирования ПС:

1. Создание эффективной и надежной системы: Целью системного проектирования является разработка программных средств, которые обеспечивают высокую производительность, надежность и безопасность при выполнении задач.

2. Удовлетворение потребностей пользователей: Системное проектирование направлено на создание ПС, которые соответствуют требованиям и ожиданиям пользователей, обеспечивая удобство использования и функциональность.

3. Масштабируемость и гибкость: Целью является создание ПС, способных масштабироваться и адаптироваться к изменяющимся потребностям бизнеса или пользователей.

4. Эффективное использование ресурсов: Системное проектирование направлено на оптимизацию использования ресурсов, таких как память, процессорное время и сетевые ресурсы.

Принципы системного проектирования ПС:

1. Модульность: Разделение системы на независимые модули для упрощения разработки, тестирования и сопровождения.

2. Интеграция: Обеспечение взаимодействия между модулями для создания целостной системы.

3. Интерфейсы: Создание четких интерфейсов между компонентами системы для упрощения их взаимодействия.

Контекст:

1. Постановка задачи

2. Анализ требований: Сбор и анализ требований пользователей для определения функциональности и характеристик системы.

*3. Проектирование архитектуры: Разработка общей структуры системы, включая определение компонентов, интерфейсов и взаимодействия между ними. Определение состава необходимого оборудования, программного обеспечения и операций, выполняемых обслуживающим персоналом.*

*Архитектурой ПО называют совокупность базовых концепций (принципов)построения программного обеспечения.*

*4. Детальное проектирование: Проектирование отдельных модулей и компонентов системы с учетом технических деталей.*

5. Разработка и интеграция: Создание кода для каждого модуля и последующая интеграция этих модулей в целостную систему.

6. Сопровождение



Также в этап проектирования входит выбор парадигмы программирования.

основные свойства требований в ТЗ:

1. Требование должно быть понятным.

2. Требование должно быть конкретным.

3. Требование должно быть тестируемым.

Если результат выполнения требования невозможно протестировать, значит, оно либо не понятное, либо не конкретное.

Техническое задание – это документ, в основе которого лежат требования, сформулированные на понятном (привычном) для Заказчика языке.

При этом должна использоваться отраслевая терминология, понятная Заказчику.

Никаких привязок к особенностям технической реализации быть не должно.

 На этапе ТЗ не важно, на какой платформе будут реализовываться эти требования.

Однако, если речь идет о внедрении системы на основе уже существующего ПО, то такая привязка может иметь место, но только на уровне экранных форм, форм отчетов и прочего. Выяснением и формулированием требований, а также разработкой Технического задания должен заниматься бизнес- аналитик, а также возможно участие программиста.

ГОСТ рекомендует следующие разделы ТЗ:

- общие сведения;

- назначение и цели создания (развития) системы;

- характеристика объектов автоматизации -- краткие сведения об объекте автоматизации или ссылки на документы, содержащие такую информацию.;

- требования к системе:

* требования к структуре и функционированию системы;
* требования к численности и квалификации персонала системы
* режиму работы системы и персонала;
* характеристики качества системы;
* требования к надежности;
* требования безопасности;
* требования к эргономике и технической эстетике;
* требования к транспортабельности для подвижных АС;
* требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
* требования к защите информации от несанкционированного доступа;
* требования по сохранности информации при авариях;
* требования к защите от влияния внешних воздействий;
* требования к патентной чистоте;
* требования по стандартизации и унификации;
* к квалификации;

- состав и содержание работ по созданию системы;

- порядок контроля и приемки системы;

- требования к составу и содержанию работ по

подготовке объекта автоматизации к вводу

системы в действие;

- требования к документированию;

- источники разработки.

1. **Задачи и особенности объектно-ориентированного проектирования программных средств. Основные стили и парадигмы программирования.**

1. Императивная

• Процедурная

• Структурная

• Модульная

• Аспектно-ориентированная

• Объектно-ориентированная

• Субъектно-ориентированная

• Агентно-ориентированная

• Компонентно-ориентированная

• Прототипно-ориентированная

• Обобщённое программирование

2. Конкатенативная

3. Декларативная

 Функциональная

 Аппликативная

 Комбинаторная

 Основанная на продолжениях

 В терминах Рефал-машины

 Логическая

 Ограничениями

 Потоком данных

4. Метапрограммирование

 Языково-ориентированная

 Пользовательская

 Автоматизация процесса программирования  Рефлексивное

5. Параллельная

6. Событийно-ориентированная

 Реактивная

7. Сервис-ориентированная

 Автоматная

Объектно-ориентированное проектирование (ООП) — это методология разработки программного обеспечения, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Основные задачи ООП:

* Создание понятной и легко поддерживаемой структуры программы;
* Повышение надёжности программного обеспечения за счёт использования абстракции, инкапсуляции и полиморфизма;
* Обеспечение возможности повторного использования кода путём создания библиотек классов.

1. **Планирование жизненного цикла программных средств. Цели и задачи планирования. Стандарты ISO 16326 и ISO 90003. Планирование процессов управления качеством сложных программных средств.**

Жизненным циклом ПО называют весь период существования ПО, связанный с подготовкой к его разработке, разработкой, использованием и переработками, начиная с того момента, когда принимается решение о его разработке до того момента, когда полностью прекращается всякое ее использование.

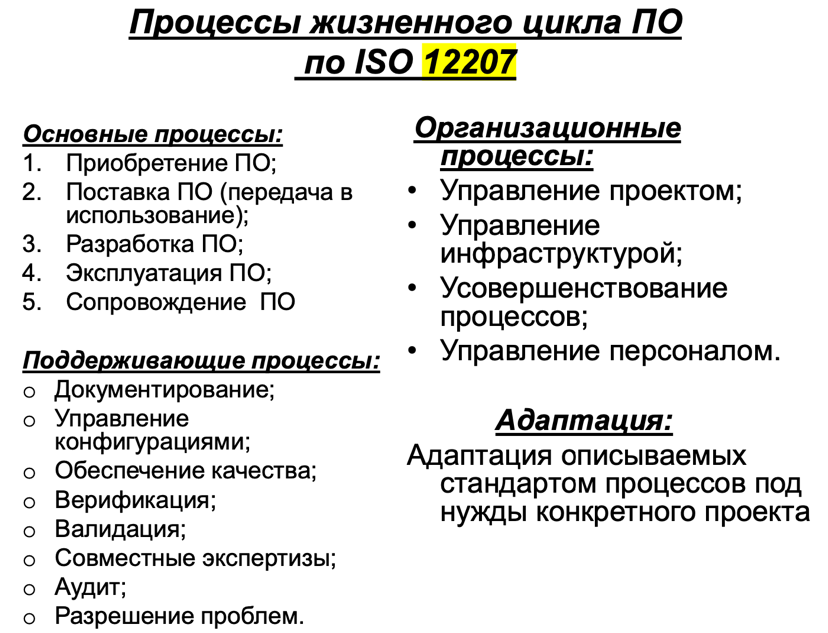
Цели планирования жизненного цикла ПС:

* Обеспечение соответствия ПС требованиям заказчика и стандартам качества.
* Снижение рисков и затрат на разработку, внедрение и сопровождение ПС.
* Повышение эффективности и результативности процесса разработки ПС.

Задачи планирования жизненного цикла ПС:

* Определение сроков и бюджета проекта.
* Распределение ресурсов между этапами проекта.
* Учёт возможных рисков и проблем.
* Контроль выполнения плана и его корректировка при необходимости.

ISO/IEC 12207 (также известный как IEEE 12207) - это международный стандарт, который определяет процессы жизненного цикла программных систем.

****

Стандарт ISO/IEC 16326 дополняет ISO/IEC 12207, предоставляя модель процессов жизненного цикла программных средств и предписывая методы их применения.

ISO 9003 — это стандарт, который устанавливает требования к системам качества для организаций, занимающихся проектированием и разработкой продукции. Он является одним из стандартов серии ISO 9000, посвящённых системам менеджмента качества.

Для планирования процессов управления создаются методологии.

Назначение методологии СММ/CMMI – системы и модели оценки зрелости – состоит в предоставлении необходимых общих рекомендаций и инструкций предприятиям, производящим ПС, по выбору стратегии совершенствования качества процессов и продуктов, путем анализа степени их производственной зрелости и оценивания факторов, в наибольшей степени влияющих на качество ЖЦ ПС, а также посредством выделения процессов, требующих модернизации.

Модель CMMI (Capability Maturity Model Integration) также представляет пять уровней зрелости, но она расширяет и дополняет модель CMM, включая не только процессы разработки программного обеспечения, но и другие аспекты организации. Вот краткое описание каждого уровня модели CMMI:

1. Уровень 1: Начальный (Performed)

- Организация имеет базовые процессы, которые выполняются по мере необходимости. Управление процессами находится на уровне индивидуальных усилий.

2. Уровень 2: Управляемый (Managed)

- Организация фокусируется на управлении проектами и процессами. Процессы документированы, стандартизированы и повторяемы.

3. Уровень 3: Определенный (Defined)

- Организация имеет установленные стандарты и процессы, которые описаны и управляются. Процессы разработки интегрированы с процессами управления организацией.

4. Уровень 4: Квантитативно управляемый (Quantitatively Managed)

- Организация использует количественные методы управления процессами для достижения улучшения качества продукта и процессов разработки.

5. Уровень 5: Оптимизируемый (Optimizing)

- Организация стремится к постоянному улучшению процессов через инновации, внедрение передовых технологий и автоматизацию.

1. **Ключевые технические характеристики основных направлений развития технологий искусственного интеллекта, которые определяют их качество и конкурентоспособность.**

Ключевые технические характеристики основных направлений развития технологий искусственного интеллекта, которые определяют их качество и конкурентоспособность:

* Точность и надёжность. Точность алгоритмов искусственного интеллекта (ИИ) определяет их способность правильно классифицировать данные и делать точные прогнозы. Надёжность алгоритмов ИИ означает их устойчивость к ошибкам и способность работать в различных условиях.
* Скорость работы. Скорость работы алгоритмов ИИ важна для их практического применения. Быстрые алгоритмы ИИ могут обрабатывать большие объёмы данных и быстро выдавать результаты.
* Масштабируемость. Масштабируемость алгоритмов ИИ означает их способность эффективно работать с большими объёмами данных. Масштабируемые алгоритмы ИИ могут быть легко адаптированы для работы с новыми данными и задачами.
* Интерпретируемость и объяснимость. Интерпретируемость алгоритмов ИИ позволяет понять, как они принимают решения. Объяснимые алгоритмы ИИ предоставляют информацию о том, почему они делают определённые выводы. Это важно для доверия к результатам ИИ и их использования в критически важных областях.
* Безопасность. Безопасность алгоритмов ИИ является ключевым аспектом их применения. Алгоритмы ИИ должны быть защищены от несанкционированного доступа, атак и других угроз безопасности.
* Адаптивность и обучаемость. Адаптивные и обучаемые алгоритмы ИИ способны адаптироваться к новым условиям и обучаться на основе новых данных. Это делает их более гибкими и эффективными.
* Интеграция с другими технологиями. Интеграция алгоритмов ИИ с другими технологиями, такими как машинное обучение, глубокое обучение и обработка естественного языка, может привести к созданию более мощных и эффективных систем.
* Энергоэффективность. Энергоэффективные алгоритмы ИИ потребляют меньше энергии при работе, что делает их привлекательными для использования в мобильных устройствах и других ограниченных по ресурсам системах.
* Совместимость с существующими системами. Совместимость алгоритмов ИИ с существующими информационными системами и платформами обеспечивает лёгкую интеграцию и использование в существующих проектах.

Эти характеристики определяют качество и конкурентоспособность алгоритмов искусственного интеллекта. Они важны для разработки и внедрения новых технологий ИИ, а также для оценки их эффективности и применимости в различных областях.

Объяснимый искусственный интеллект (Explainable AI, XAI) – модель, которая могла бы в перспективе объяснять механизмы, лежащие за алгоритмами машинного обучения.

Множество решений, применяющих алгоритмы ИИ, представляют собой подобие «черного ящика», – зачастую не только конечные пользователи, но и сами разработчики не могут точно определить, как именно модель машинного обучения пришла к тем или иным выводам в ходе обработки исходных данных.

1. **Методология обеспечения качества ПС в программной инженерии. Базовые принципы стандартов качества программного обеспечения в соответствии со стандартами ISO 9000:2000 и ISO 15504:1-9.**

Общее представление о качестве ПС международным стандартом ISO 9126:1-4:2002 описывают 3 взаимодействующими метриками качества, отражающими:

- внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;

- внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;

- качество при использовании в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.

Модель характеристик качества ПС и компонентов состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками

- пригодность применения (защищенность, соответствие стандартам, способность квзаимодействию)

- надежность

- применимость (понятность, простота использования)

- эффективность

- сопровождаемость

- переносимость

Серия стандартов ISO 9000:2000 разработана, чтобы помочь предприятиям всех типов и размеров внедрить и использовать эффективные системы менеджмента (административного управления) качества.

1.ISO 9000:2000 – представляет введение в системы управления качеством продукции и услуг и словарь качества;

2.ISO 9001:2000 – устанавливает детальные требования для систем управления качеством, достаточные в случае необходимости продемонстрировать способность предприятия, обеспечить соответствие качества продукции и услуг требованиям заказчика;

3.ISO 9004:2000 – содержит руководство по внедрению и применению широко развитой системы управления качеством, чтобы достичь постоянного улучшения деловой деятельности и результатов предприятия.

Стандарт ISO 15504:1-5:2003-2006 регламентирует оценку и аттестацию зрелости процессов создания, сопровождения и совершенствования программных средств и систем, выполняемых предприятиями. Применение стандарта ISO 15504:1-5:2003-2006 направлено на выработку предприятиями и специалистами культуры постоянного совершенствования зрелости технологий обеспечения ЖЦ ПС, отвечающих бизнес-целям проектов и оптимизации использования доступных ресурсов.

1. **Организация документирования программных средств. Типы документаций.**

Документация является органической, составной частью программного продукта для ЭВМ и требуются значительные ресурсы для ее создания и применения.Тексты и объектный код программ для ЭВМ могут стать программным продуктом только в совокупности с комплексом документов, полностью соответствующих их содержанию и достаточных для его освоения, применения и изменения.

Документы должны быть:

* корректными,
* строго адекватными текстам программ и содержанию БД,
* систематически, структурировано и понятно изложены

Процессы документирования программ и данных входят в весь ЖЦ сложных систем и ПС. Поэтому организация и реализация работ по созданию документов должны распределяться между специалистами, ведущими непосредственное и преимущественное создание проектов комплексов.

При создании особо сложных систем целесообразно выделение специального коллектива, обеспечивающего организацию и реализацию основных системных работ по документообороту ПС.

Совокупные затраты на документирование крупных программных продуктов могут достигать 20 – 30% от общей трудоемкости проекта и необходимого числа (десятки) специалистов в жизненном цикле проекта ПС.

По своему назначению и ориентации на определенные задачи и группы пользователей, документацию ПС можно разделить:

* Технологическую документацию процессов разработки и обеспечения всего ЖЦ, включающую подробные технические описания.

Обеспечивает возможность отчуждения, детального освоения, развития и корректировки ими программ и данных на всем жизненном цикле ПС;

* Эксплуатационную документацию объекта и результатов разработки, создаваемую для конечных пользователей ПС и позволяющую им осваивать и квалифицированно применять эти средства для решения конкретных функциональных задач систем.

Эксплуатационная документация должна обеспечивать отчуждаемость программного продукта от первичных разработчиков и возможность освоения и эффективного применения комплексов программ достаточно квалифицированными специалистами – пользователями ПС и системы.

1. **Методы тестирования программных систем. Процессы и средства тестирования программных компонентов.**

Тестирование ПС от требований имеет две взаимодополняющие цели:

* Показать, что ПС удовлетворяет заданным требованиям к нему.
* Показать с высокой степенью доверия, что устранены дефекты и ошибки, которые могли бы привести к возникновению отказов, влияющих на корректность и надежность системы.

При первой стратегии тестирования за основу принимается структура ПМ, построенная по тексту программы в виде графа (УГП).

1. В УПГ выделяются и упорядочиваются маршруты исполнения программ.
2. Эти условия используются для подготовки тестовых наборов, каждый из которых должен реализоваться по маршруту, принятому за эталон при подготовке теста.
3. Отклонение исполнения теста от первоначально выбранного маршрута рассматривается как ошибка

При второй стратегии за основу принимаются требования спецификаций, конкретные тестовые и эталонные значения.

1. При каждом тесте программа исполняется по определенному маршруту, который регистрируется.
2. По мере тестирования отмечаются проверенные операторы, и оценивается полнота покрытия тестами требований спецификаций на маршрутах тестирования

Программы в процессе тестирования используются в двух различных

формах представления:

1. текст программы на языке программирования или формализованного описания спецификаций требований (графическое представление);

2. Текст в машинном коде конкретной ЭВМ (объектное представление).

Системы автоматизации тестирования можно разделить на два вида:

* статические системы, которые анализируют спецификации и исходные тексты программ в объектном коде без их исполнения на ЭВМ (анализаторы);
* Динамические системы, при использовании которых программы функционируют в объектном коде и пригодны для их реального применения на ЭВМ.

В группу средств статической отладки входят также средства расчета длительностей исполнения модулей и компонентов. В результате расчетов выявляются компоненты программы, требующие избыточно большого времени счета на реализующей ЭВМ.

Группу средств динамической отладки можно разделить на два типа:

1. Основные средства, непосредственно обеспечивающие исполнение программ в соответствии с отладочными заданиями (транслятор, исполнение по сценарию, регистрация результатов).

2. Вспомогательные средства, которые анализируют выполненное тестирование, его результаты и проведенные корректировки.

1. **Управление ресурсами в жизненном цикле программных средств (ПС). Ресурсы для обеспечения функциональной пригодности при разработке сложных ПС.**

Управление ресурсами в жизненном цикле программных средств (ПС) представляет собой процесс планирования, распределения и контроля ресурсов, необходимых для разработки, внедрения и сопровождения программного обеспечения.

Ресурсы для обеспечения функциональной пригодности при разработке сложных ПС включают:

* Человеческие ресурсы: разработчики, тестировщики, аналитики, менеджеры проектов и другие специалисты, необходимые для создания и поддержки программного продукта.
* Финансовые ресурсы: бюджет на разработку, тестирование, внедрение и сопровождение программного средства.
* Технические ресурсы: оборудование (компьютеры, серверы, сети), программное обеспечение (операционные системы, инструменты разработки), а также лицензии на использование сторонних компонентов.
* Информационные ресурсы: данные, необходимые для разработки и тестирования программного продукта, включая спецификации, требования, тестовые сценарии и результаты тестирования.
* Временные ресурсы: сроки выполнения этапов жизненного цикла программного средства, а также время, необходимое для устранения дефектов и внесения изменений в продукт.
* Интеллектуальные ресурсы: знания и опыт разработчиков, а также методология и стандарты, используемые при разработке программного продукта.
* Материальные ресурсы: расходные материалы, такие как бумага, картриджи для принтеров и т. д., необходимые для документирования и сопровождения программного средства.
* Организационные ресурсы: структура управления проектом, роли и обязанности участников проекта, а также процедуры и процессы, используемые для разработки и сопровождения программного продукта.
* Правовые ресурсы: лицензии, авторские права и другие юридические документы, связанные с разработкой и использованием программного продукта.

Для эффективного управления ресурсами необходимо провести анализ требований к программному продукту, определить необходимые ресурсы и разработать план их использования. Также важно контролировать выполнение плана и вносить коррективы при необходимости. Это позволит обеспечить высокое качество и функциональную пригодность программного продукта, а также снизить риски и затраты на его разработку и сопровождение.

1. **Дефекты, ошибки и риски в жизненном цикле программных средств. Типы ошибок и дефектов. Риски в жизненном цикле сложных программных средств.**

Статистика ошибок и дефектов в комплексах программ и их характеристики в конкретных типах проектов ПС, могут служить ориентирами для разработчиков при распределении ресурсов в жизненном цикле ПС и предохранять их от излишнего оптимизма при оценке достигнутого качества программных продуктов.

К понятию риски относятся негативные события и их величины, отражающие потери, убытки или ущерб от процессов или продуктов, вызванные дефектами при проектировании требований, недостатками обоснования проектов ПС, а также при последующих этапах разработки, реализации и всего жизненного цикла комплексов программ.

в общем случае под ошибкой подразумевается неправильность, погрешность или неумышленное искажение объекта или процесса, что может быть причиной ущерба – риска при функционировании и применении программы.

Классификация ошибок

\*Небольшими ошибками называют такие, на которые средний пользователь может и не обратить внимания при применении ПС. \*Небольшие ошибки могут включать орфографические ошибки на экране, пропущенные разделы в справочнике и другие мелкие проблемы.

\*По десятибалльной шкале рисков небольшие ошибки находятся в пределах от 1 до 3 приоритета.

Умеренными ошибками называют те, которые влияют на конечного пользователя, но имеются слабые последствия или обходные пути, позволяющие сохранить достаточную функциональность ПС.

К умеренным ошибкам относятся дефекты:

неверные ссылки на страницах,

 ошибочный текст на экране,

сбои, если их трудно воспроизвести, и они не

оказывают влияния на существенное число

пользователей.

Ошибки, которые можно исправить на этом уровне,

следует исправлять, если на это есть время и

возможность.

По десятибалльной шкале умеренные ошибки находятся в диапазоне от 4 до 7 приоритета.

Критические ошибки останавливают выпуск версии программного продукта!

Это могут быть ошибки с высоким влиянием, которые вызывают сбой в системе или потерю данных, отражаются на надежности и безопасности применения ПС, с которыми никогда не передается комплекс программ пользователю.

По десятибалльной шкале – от 8 до 10 приоритета.

Причины возникновения и проявления рисков:

1. Злоумышленные, активные воздействия заинтересованных лиц.

Для решения этой проблемы созданы и активно развиваются методы, средства и стандарты обеспечения защиты программ и данных от предумышленных негативных внешних воздействий.

2. Риски при случайных, дестабилизирующих воздействиях дефектов ПС и отсутствии предумышленного негативного влияния на системы, ПС или информацию баз данных.

Для устранения или снижения рисков до допустимых пределов может быть необходимо изменение требований к функциональной пригодности, к конструктивным характеристикам и доступным ресурсам.

1. **Процессы управления конфигурацией программного обеспечения в соответствии со стандартами ISO 12207 и ISO 15846.**

Стандарт ISO 15846 обобщает, детализирует и развивает основные концептуальные положения в части управления конфигурацией, представленные в стандарте ISO 12207.

**управление конфигурацией** -- обеспечение управляемого и контролируемого развития структуры ПС, состава компонентов и функций, а также сокращение дефектов в течение всего жизненного цикла ПС.

**Процессы управления конфигурацией ПС направлены на достижения следующих основных целей:**

- обеспечить возможность оценки соответствия требованиям заказчика результатов жизненного цикла программного средства;

- обеспечить управляемую конфигурацию ПС на протяжении всего ЖЦ ;

- обеспечить управление входными и выходными данными процесса в течение всего ЖЦ, что гарантирует непротиворечивость и повторяемость работ в процессах;

- обеспечить контрольные точки для проверки, оценки состояния и контроля изменений посредством управления элементами конфигурации и определения базовой версии программного продукта;

- обеспечить контроль над тем, чтобы фиксировались дефекты и ошибки, а изменения регистрировались, утверждались и реализовались;

- гарантировать надежное физическое архивирование, восстановление и сопровождение единиц конфигурации ПС.

**Управление конфигураций включает следующие задачи:**

* подготовку процесса;
* определение конфигурации;
* контроль конфигурации;
* учет состояний конфигурации;
* оценку конфигурации;
* управление выпуском и поставку программного продукта.

**План изменений конфигурации должен содержать следующие разделы:**

- почему и с какой целью производится корректировка программ или данных;

- кто выполняет, и кто санкционирует проведение изменений комплекса программ или компонентов;

- какие действия и процедуры должны быть выполнены для реализации изменений единиц конфигурации;

- когда по срокам и в координации, с какими другими процедурами следует реализовать определенную модификацию компонентов и конфигурацию ПС;

- как и с использованием, каких средств и ресурсов должны быть выполнены запланированные изменения ПС и компонентов.

1. **Методика оформления отчетов о выявленных дефектах, ошибках и предложениях по корректировке версий ПС.**

Методика оформления отчётов о выявленных дефектах, ошибках и предложениях по корректировке версий программных средств (ПС)

* Идентификация проблемы:
* Определить тип проблемы: дефект, ошибка или предложение по улучшению.
* Дать краткое описание проблемы, включая её влияние на функциональность ПС.
* Сбор информации:
* Собрать все необходимые данные для описания проблемы, такие как скриншоты, логи, трассировки стека и т. д.
* Зафиксировать шаги для воспроизведения проблемы.
* Классификация проблемы:
* Классифицировать проблему в соответствии с системой отслеживания ошибок (например, по приоритету, типу, категории).
* Присвоить уникальный идентификатор проблеме.
* Описание проблемы:
* Подробно описать проблему, включая все детали, которые могут помочь в её понимании и решении.
* Указать ожидаемое поведение и фактическое поведение.
* Влияние на функциональность:
* Оценить влияние проблемы на функциональность и производительность ПС.
* Рассмотреть возможные последствия для пользователей.
* Предложения по решению:
* Предложить возможные решения проблемы.
* Обосновать каждое предложенное решение.
* Оценка сложности:
* Оценить сложность реализации каждого предложенного решения.
* Предположить время, необходимое для реализации.
* Приоритет:
* Установить приоритет проблемы в зависимости от её влияния на функциональность, производительность и безопасность.
* Документация:
* Подготовить отчёт о проблеме, включающий все собранные данные, описание проблемы, предложения по решению и оценку сложности.
* Отправка отчёта:
* Отправить отчёт в систему отслеживания ошибок для дальнейшего рассмотрения и обработки.
* Отчёт должен быть ясным, кратким и содержать всю необходимую информацию для понимания проблемы и её решения. Он должен помочь разработчикам быстро и эффективно исправить дефекты и ошибки в ПС.

1. **Интеграция, квалификационное тестирование и испытания комплексов программ. Подходы для интеграционного тестирования комплексов программ.**

Квалификационное тестирование -- тестирование, проводимое разработчиком и санкционированное приобретающей стороной (при необходимости) с целью демонстрации того, что программный продукт удовлетворяет спецификациям и готов для применения в заданном окружении или интеграции с системой, для которой он предназначен.

Интеграционное тестирование комплексов программ - это процесс проверки взаимодействия между различными компонентами программного комплекса для обеспечения их корректной работы вместе.

**Подходы для интеграционного тестирования комплексов программ:**

- Сверху-вниз (Top-Down): При этом подходе тестирование начинается с верхних уровней системы и постепенно опускается к нижним. Это позволяет выявить проблемы взаимодействия между компонентами раньше, чем если бы тестирование начиналось с нижних уровней.

- Снизу-вверх (Bottom-Up): В отличие от верхнего подхода, здесь тестирование начинается с низших уровней системы и постепенно продвигается к верхним. Это позволяет выявить проблемы на уровне компонентов и модулей до их интеграции в систему.

- Покомпонентный (Component Stubs): Здесь тестируются отдельные компоненты системы с использованием заглушек (stubs) для имитации поведения других компонентов. Это позволяет проверить работоспособность отдельных частей системы независимо от других компонентов.

1. **Процессы эксплуатации и сопровождения ПС в жизненном цикле. Этапы и процедуры при сопровождении программных средств в соответствии с требованиями стандарта ISO 12207 по развитию и модификации программного продукта в жизненном цикле.**

Процессы эксплуатации и сопровождения ПС в жизненном цикле

Жизненный цикл программного средства (ПС) включает в себя несколько этапов, одним из которых является эксплуатация и сопровождение. Эти процессы направлены на обеспечение надёжной и эффективной работы программного продукта после его внедрения.

Эксплуатация ПС — это процесс использования программного продукта пользователями для выполнения своих задач. В ходе эксплуатации могут возникать различные проблемы, связанные с работой программы, которые необходимо решать.

Сопровождение ПС — это комплекс мероприятий, направленных на поддержание работоспособности программного продукта в течение всего срока его эксплуатации. Сопровождение включает в себя следующие основные задачи:

основная задача сопровождения - изменить и улучшить существующий программный продукт, сохраняя его целостность и функциональную пригодность.

Работы по сопровождению ПС:

\*подготовка процесса;

\*анализ проблем и изменений;

\*внесение изменений;

\*проверка и приемка при сопровождении;

\*перенос на иные платформы;

\*снятие с эксплуатации.

Общий план сопровождения должен определять:

- причины сопровождения;

- состав исполнителей работ;

- роли и обязанности каждого субъекта, вовлеченного в сопровождение;

- как должны быть выполнены основные процессы;

- какие имеются и какие необходимы ресурсы для сопровождения;

- методы и средства организации работ по управлению, выпуску продукта и синхронизации работ;

- перечень всех проектных результатов и продуктов, подлежащих поставке заказчику;

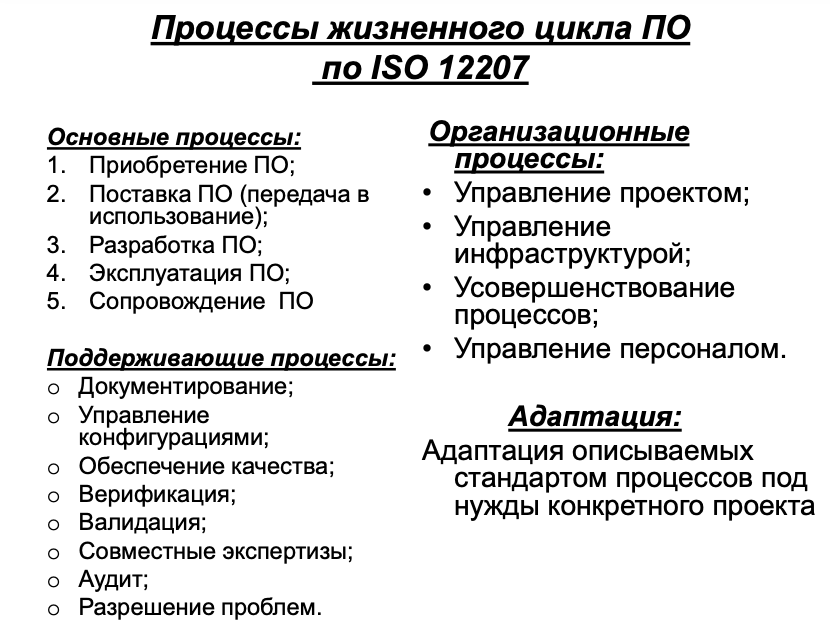
- критерии завершения соответствующей деятельности, работ и задач;

- состав отчетных материалов по этапам, затратам и графикам проведения работ;

- периодичность и способы выдачи отчетных материалов;

- состав отчетных материалов по проблемам и устраненным дефектам;

- время начала и длительность сопровождения.

****

1. **Характеристики качества баз данных. Модели качества по стандарту ISO 9126:1-4:2002.**

Качество базы данных — это совокупность свойств, которые определяют её способность удовлетворять требованиям пользователей и обеспечивать эффективное функционирование системы управления базой данных.

Основные характеристики качества баз данных:

* Функциональность: способность базы данных выполнять требуемые функции в соответствии с требованиями пользователей.
* Надёжность: свойство базы данных сохранять работоспособность при сбоях и отказах оборудования, ошибках пользователей и других неблагоприятных условиях.
* Производительность: время отклика на запросы пользователей, скорость обработки транзакций и другие показатели эффективности работы базы данных.
* Безопасность: защита данных от несанкционированного доступа, изменения или уничтожения.
* Масштабируемость: возможность расширения базы данных для удовлетворения растущих потребностей пользователей без существенного снижения производительности.
* Удобство использования: простота и понятность интерфейса пользователя, лёгкость обучения и работы с базой данных.
* Сопровождаемость: лёгкость внесения изменений в структуру базы данных, добавления новых функций и устранения ошибок.
* Совместимость: возможность интеграции с другими системами и платформами.
* Переносимость: возможность переноса базы данных на другую платформу или операционную систему без существенных изменений.

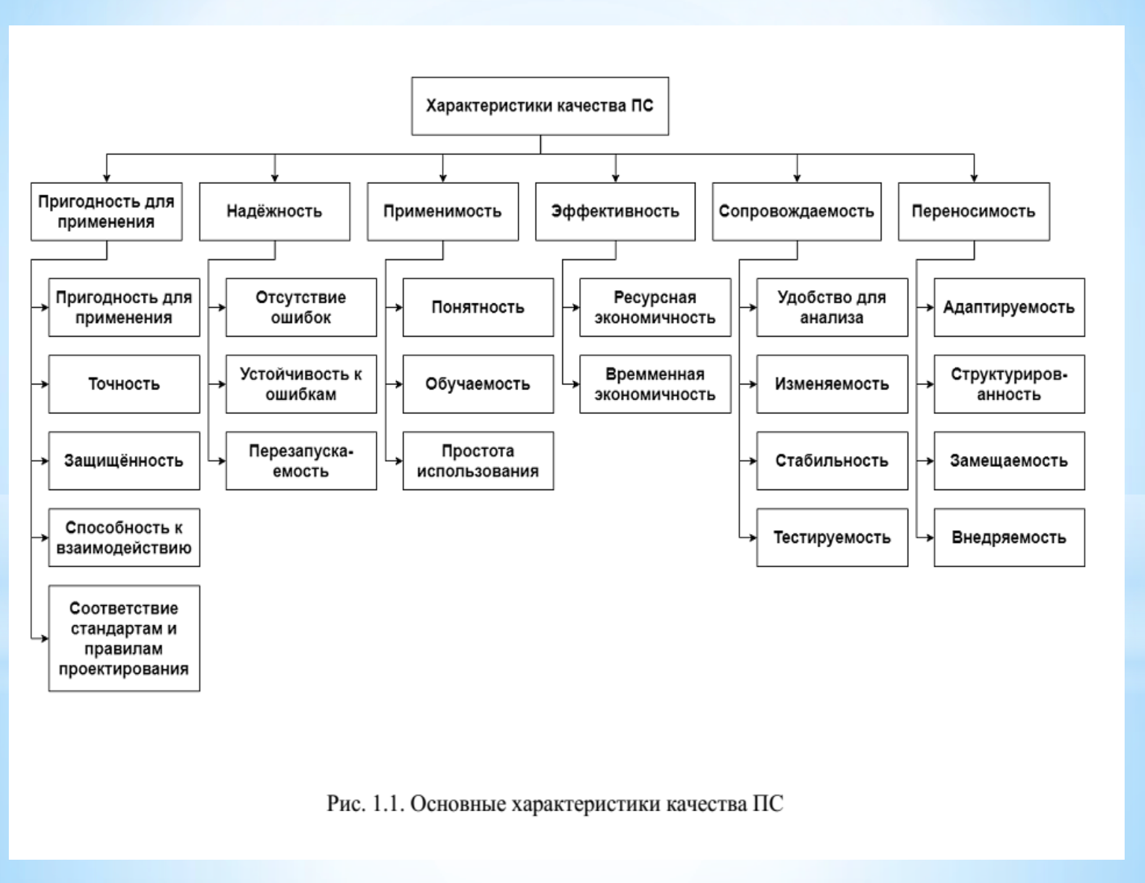
Общее представление о качестве ПС международным стандартом ISO 9126:1-4:2002 описывают 3 взаимодействующими метриками качества, отражающими:

- внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов жизненного цикла ПС;

- внешнее качество, заданное требованиями заказчика в спецификациях и отражающееся характеристиками конечного продукта;

- качество при использовании в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.

Модель характеристик качества ПС и компонентов состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками:



1. **Основные парадигмы программирования. Задачи и особенности структурного, объектно-ориентированного и мультиагентного проектирования сложных систем.**

1. Императивная

• Процедурная

• Структурная

• Модульная

• Аспектно-ориентированная

• Объектно-ориентированная

• Субъектно-ориентированная

• Агентно-ориентированная

• Компонентно-ориентированная

• Прототипно-ориентированная

• Обобщённое программирование

2. Конкатенативная

3. Декларативная

 Функциональная

 Аппликативная

 Комбинаторная

 Основанная на продолжениях

 В терминах Рефал-машины

 Логическая

 Ограничениями

 Потоком данных

4. Метапрограммирование

 Языково-ориентированная

 Пользовательская

 Автоматизация процесса программирования  Рефлексивное

5. Параллельная

6. Событийно-ориентированная

 Реактивная

7. Сервис-ориентированная

 Автоматная

Объектно-ориентированное проектирование (ООП) — это методология разработки программного обеспечения, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Основные задачи ООП:

* Создание понятной и легко поддерживаемой структуры программы;
* Повышение надёжности программного обеспечения за счёт использования абстракции, инкапсуляции и полиморфизма;
* Обеспечение возможности повторного использования кода путём создания библиотек классов.

Структурное проектирование сложных систем представляет собой процесс разработки структуры системы, которая будет удовлетворять требованиям к её функциональности, надёжности, безопасности и другим характеристикам.

Основные задачи структурного проектирования:

* Определение состава компонентов системы и их взаимодействия;
* Разработка архитектуры системы, включая выбор технологий и инструментов разработки;
* Обеспечение надёжности и безопасности системы путём использования методов и средств защиты информации;
* Тестирование системы на соответствие требованиям и устранение выявленных дефектов.

Особенности структурного проектирования:

* Использование иерархического подхода к разработке системы, при котором компоненты нижнего уровня объединяются в более крупные блоки;
* Применение методов декомпозиции для разбиения системы на отдельные модули, каждый из которых выполняет определённую функцию;
* Создание модели системы, описывающей её структуру и поведение;
* Проведение анализа и оптимизации модели с целью улучшения характеристик системы.

Мультиагентное проектирование сложных систем — это подход к проектированию, основанный на использовании агентов как основных элементов системы. Агенты могут быть автономными или взаимодействовать друг с другом для достижения общей цели.

Задачи мультиагентного проектирования:

* Разделение сложной системы на множество агентов, каждый из которых отвечает за выполнение определённой функции;
* Организация взаимодействия между агентами для координации их действий и обмена информацией;
* Управление поведением агентов с помощью правил и стратегий;
* Адаптация системы к изменяющимся условиям среды путём обучения агентов новым правилам и стратегиям.

Особенности мультиагентного проектирования:

* Возможность создания гибких и адаптивных систем, способных реагировать на изменения внешней среды;
* Повышение эффективности системы за счёт распределения задач между агентами и параллельного выполнения операций;
* Снижение сложности системы благодаря разделению её на более простые компоненты;
* Улучшение надёжности системы за счёт использования агентов для обнаружения и устранения ошибок.

1. **История и основные направления развития искусственного интеллекта. Национальные стандарты на создание систем ИИ.**

История развития искусственного интеллекта:

Искусственный интеллект (ИИ) — это область компьютерных наук, которая занимается разработкой интеллектуальных агентов, способных выполнять задачи, требующие человеческого интеллекта. ИИ включает в себя множество направлений, таких как машинное обучение, глубокое обучение, обработка естественного языка, компьютерное зрение и робототехника.

* 1950-е годы: Начало исследований в области искусственного интеллекта. Первые работы были посвящены разработке алгоритмов для решения задач, требующих человеческого интеллекта, таких как игра в шахматы и доказательство теорем.
* 1960-е годы: Появление первых экспертных систем, которые могли решать задачи в определённых областях знаний.
* 1970-е годы: Развитие методов машинного обучения, которые позволяли компьютерам обучаться на основе данных.
* 1980-е годы: Рост интереса к искусственному интеллекту после появления персональных компьютеров. Разработка новых алгоритмов и методов, таких как нейронные сети и генетические алгоритмы.
* 1990-е годы: Внедрение методов глубокого обучения, которые позволили компьютерам учиться на больших объёмах данных.
* 2000-е годы: Быстрый рост популярности искусственного интеллекта благодаря развитию технологий обработки больших данных и появлению новых алгоритмов.
* Настоящее время: Искусственный интеллект становится всё более распространённым и используется во многих областях, таких как медицина, финансы, транспорт и производство.

Основные направления развития искусственного интеллекта включают:

* Машинное обучение: Алгоритмы, которые позволяют компьютерам обучаться на данных без явного программирования. Включает в себя методы обучения с учителем, обучения без учителя и обучения с подкреплением.
* Глубокое обучение: Методы машинного обучения, основанные на нейронных сетях, которые могут обучаться на больших объёмах данных и решать сложные задачи.
* Обработка естественного языка: Алгоритмы для понимания и генерации естественного языка. Включает в себя распознавание речи, машинный перевод и генерацию текста.
* Компьютерное зрение: Алгоритмы для анализа изображений и видео. Включает в себя обнаружение объектов, распознавание лиц и анализ сцен.
* Робототехника: Применение искусственного интеллекта в роботах для выполнения задач, требующих восприятия, планирования и действия.
* Автоматизация процессов: Использование искусственного интеллекта для автоматизации рутинных задач, таких как управление производственными процессами и обслуживание клиентов.

Национальные стандарты на создание систем искусственного интеллекта

В России и других странах существуют национальные стандарты, которые регулируют разработку и использование систем искусственного интеллекта (ИИ). Эти стандарты направлены на обеспечение безопасности, надёжности и этичности применения ИИ в различных областях.

В России разработкой стандартов в области искусственного интеллекта занимается технический комитет по стандартизации № 164 «Искусственный интеллект», созданный на базе Российского института стандартизации (Росстандарт). Комитет разрабатывает стандарты для различных аспектов ИИ, таких как терминология, методы оценки качества, безопасность и этические аспекты.

Некоторые из основных стандартов, разработанных комитетом:

* ГОСТ Р 59277-2020 «Системы искусственного интеллекта. Классификация систем искусственного интеллекта»;
* ГОСТ Р ИСО/МЭК 23894-1–2023 «Информационные технологии. Искусственный интеллект. Языки программирования для искусственного интеллекта. Часть 1. Общие положения»;
* ГОСТ Р 70519–2022 «Системы управления проектами. Системы управления проектами с применением технологий искусственного интеллекта».
* Эти стандарты определяют требования к системам ИИ, их классификации, методам разработки, тестирования и оценки. Они также устанавливают правила использования систем ИИ в разных областях, включая здравоохранение, транспорт, производство и другие.

1. **Описать причины возникновения рисков в жизненном цикле программных средств. Описать мероприятия по предотвращению рисков при создании любых сложных систем.**

Статистика ошибок и дефектов в комплексах программ и их характеристики в конкретных типах проектов ПС, могут служить ориентирами для разработчиков при распределении ресурсов в жизненном цикле ПС и предохранять их от излишнего оптимизма при оценке достигнутого качества программных продуктов.

К понятию риски относятся негативные события и их величины, отражающие потери, убытки или ущерб от процессов или продуктов, вызванные дефектами при проектировании требований, недостатками обоснования проектов ПС, а также при последующих этапах разработки, реализации и всего жизненного цикла комплексов программ.

Причины возникновения и проявления рисков:

1. Злоумышленные, активные воздействия заинтересованных лиц.

Для решения этой проблемы созданы и активно развиваются методы, средства и стандарты обеспечения защиты программ и данных от предумышленных негативных внешних воздействий.

2. Риски при случайных, дестабилизирующих воздействиях дефектов ПС и отсутствии предумышленного негативного влияния на системы, ПС или информацию баз данных.

Для устранения или снижения рисков до допустимых пределов:

Подготовка исходных данных:

* определение целей, назначения и функций проекта программного средства и системы;
* разработка предварительных требований к функциональной пригодности и конструктивным характеристикам ПС;
* формирование группы экспертов для анализа угроз и управления рисками

Выделение, идентификация, анализ угроз и рисков программного средства:

* выделение источников и угроз нарушения требований и ограничений ресурсов, определение критериев работоспособности проекта ПС;
* анализ причин, выделение категорий угроз и возможных последствий рисков функциональной пригодности ПС;
* анализ причин, угроз и возможных рисков конструктивных характеристик и ограничения ресурсов проекта ПС

Оценивание опасности угроз и рисков и выбор контрмер для их сокращения:

* оценивание возможных последствий, уровней потенциальных угроз и приоритетов категории рисков проекта ПС;
* планирование методов и ресурсов реализации контрмер для сокращения опасных, приоритетных рисков проекта ПС;
* распределение ресурсов на контрмеры для сбалансированного сокращения интегрального риска проекта ПС

Сокращение или ликвидация опасных рисков пс:

* реализация контрмер для сокращения интегрального риска и составление отчетов о состоянии проекта;
* корректировка требований к функциональной пригодности, конструктивным характеристикам и ресурсов проекта ПС;
* регистрация результатов сокращения интегрального риска на очередном этапе проекта ПС

Контроль, регистрация, мониторинг и утверждение допустимого интегрального риска пс:

* контроль, отслеживание и мониторинг реализации сокращения интегрального риска по этапам проекта ИС;
* документирование и утверждение допустимого интегрального риска по этапам жизненного цикла проекта ПС;
* оформление итогового отчета по результатам сокращения рисков ПС

1. **Процессы управления конфигурацией программного обеспечения в соответствии со стандартами ISO 12207 и ISO 15846.**

Стандарт ISO 15846 обобщает, детализирует и развивает основные концептуальные положения в части управления конфигурацией, представленные в стандарте ISO 12207.

**управление конфигурацией** -- обеспечение управляемого и контролируемого развития структуры ПС, состава компонентов и функций, а также сокращение дефектов в течение всего жизненного цикла ПС.

**Процессы управления конфигурацией ПС направлены на достижения следующих основных целей:**

- обеспечить возможность оценки соответствия требованиям заказчика результатов жизненного цикла программного средства;

- обеспечить управляемую конфигурацию ПС на протяжении всего ЖЦ ;

- обеспечить управление входными и выходными данными процесса в течение всего ЖЦ, что гарантирует непротиворечивость и повторяемость работ в процессах;

- обеспечить контрольные точки для проверки, оценки состояния и контроля изменений посредством управления элементами конфигурации и определения базовой версии программного продукта;

- обеспечить контроль над тем, чтобы фиксировались дефекты и ошибки, а изменения регистрировались, утверждались и реализовались;

- гарантировать надежное физическое архивирование, восстановление и сопровождение единиц конфигурации ПС.

**Управление конфигураций включает следующие задачи:**

* подготовку процесса;
* определение конфигурации;
* контроль конфигурации;
* учет состояний конфигурации;
* оценку конфигурации;
* управление выпуском и поставку программного продукта.

**План изменений конфигурации должен содержать следующие разделы:**

- почему и с какой целью производится корректировка программ или данных;

- кто выполняет, и кто санкционирует проведение изменений комплекса программ или компонентов;

- какие действия и процедуры должны быть выполнены для реализации изменений единиц конфигурации;

- когда по срокам и в координации, с какими другими процедурами следует реализовать определенную модификацию компонентов и конфигурацию ПС;

- как и с использованием, каких средств и ресурсов должны быть выполнены запланированные изменения ПС и компонентов.