Оглавление

[1. Понятие жизненного цикла программных средств. Модели жизненного цикла. 2](#_Toc11284071)

[2. Методы тестирования программных систем. Процессы и средства тестирования программных компонентов. 3](#_Toc11284072)

[3. Проблемы современной программной инженерии. Понятие сложной системы. Типы сложных систем. 5](#_Toc11284073)

[4. Интеграция, квалификационное тестирование и испытания комплексов программ. Подходы для интеграционного тестирования комплексов программ. 6](#_Toc11284074)

[5. Профили стандартов жизненного цикла систем и программных средств в области программной инженерии. Профиль жизненного цикла ПС и БД на основе стандарта ISO 12207. 6](#_Toc11284075)

[6. 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. 8](#_Toc11284076)

[7. 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. 9](#_Toc11284077)

[8. Управление конфигурацией в жизненном цикле программных средств. Планы, задачи и цели управления конфигурацией ПС. Отчёты о состоянии конфигурации. 10](#_Toc11284078)

[9. Базовые характеристики и субхарактеристики качества программных средств, описанные в стандарте ISO 9126:1-4:2002. 11](#_Toc11284079)

[10. 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. 12](#_Toc11284080)

[11. Управление проектами программных средств в системе СMMI 13](#_Toc11284081)

[12. Оценивание надежности и безопасности функционирования сложных программных средств. Оценивание эффективности использования ресурсов ЭВМ. 14](#_Toc11284082)

[13. Стандарт ISO 14252:1996. Руководство по POSIX окружению открытых систем. Понятие открытой системы. Концепция стандартов POSIX, её цели и задачи. 15](#_Toc11284083)

[14. Основные факторы, определяющие качество сложных программных средств. Метрики характеристик качества, описанные в стандарте ISO 9126:1-4:2002. 16](#_Toc11284084)

[15. Понятие сложной системы. Цели и принципы системного проектирования сложных программных средств. Процессы системного проектирования программных средств. 18](#_Toc11284085)

[16. Организация документирования программных средств. Типы документаций. 20](#_Toc11284086)

[17. 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). 20](#_Toc11284087)

[18. Верификация, тестирование и оценивание корректности программных компонентов. Цели, задачи и назначение. Виды и методы тестирования. 22](#_Toc11284088)

[19. Методика оформления отчетов о выявленных дефектах, ошибках и предложениях по корректировке версий ПС. 23](#_Toc11284089)

[20. Цели и процессы технико-экономического обоснования проектов программных средств. Методика 1 – экспертное технико-экономическое обоснование проектов программных средств. 25](#_Toc11284090)

[21. Планирование жизненного цикла программных средств. Цели и задачи планирования. Стандарты ISO 16326 и ISO 90003. Планирование процессов управления качеством сложных программных средств. 26](#_Toc11284091)

[22. Организация разработки требований к сложным программным средствам. Процессы разработки требований к характеристикам сложных программных систем. 28](#_Toc11284092)

[23. Дефекты, ошибки и риски в жизненном цикле программных средств. Типы ошибок и дефектов. Риски в жизненном цикле сложных программных средств. 29](#_Toc11284093)

[24. Задачи и особенности объектно-ориентированного проектирования программных средств. Основные стили и парадигмы программирования. 30](#_Toc11284094)

[25. Управление ресурсами в жизненном цикле программных средств (ПС). Ресурсы для обеспечения функциональной пригодности при разработке сложных ПС. 32](#_Toc11284095)

[26. Процессы управления конфигурацией программного обеспечения в соответствии со стандартами ISO 12207 и ISO 15846. 33](#_Toc11284096)

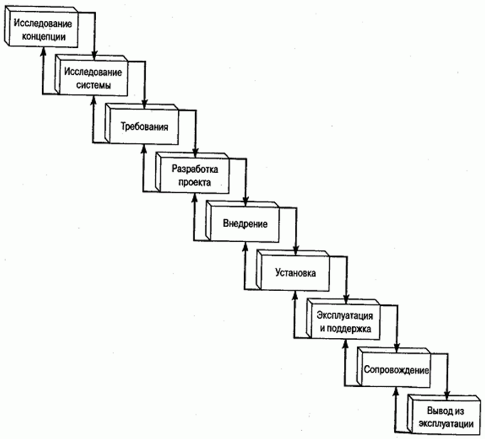
[27. Характеристики качества баз данных. Модели качества по стандарту ISO 9126:1-4:2002. 34](#_Toc11284097)

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

Термином **жизненный цикл** (ЖЦ) принято отражать совокупность процессов и этапов развития организмов живой природы, технических систем, продуктов производства от моментов зарождения или появления потребности их создания и использования до прекращения функционирования или применения.

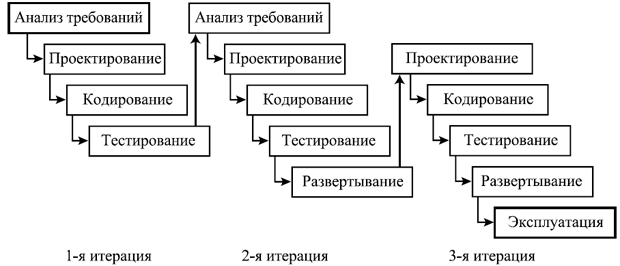
Общую модель жизненного цикла сложной системы обычно разделяют на следующие основные этапы с последующей адаптацией каждого из них в модели жизненного цикла конкретной системы:

* определение потребностей;
* исследование и описание основных концепций;
* проектирование и разработка;
* испытания системы;
* создание и производство;
* распространение и продажа;
* эксплуатация;
* сопровождение и мониторинг;
* снятие с эксплуатации (утилизация).

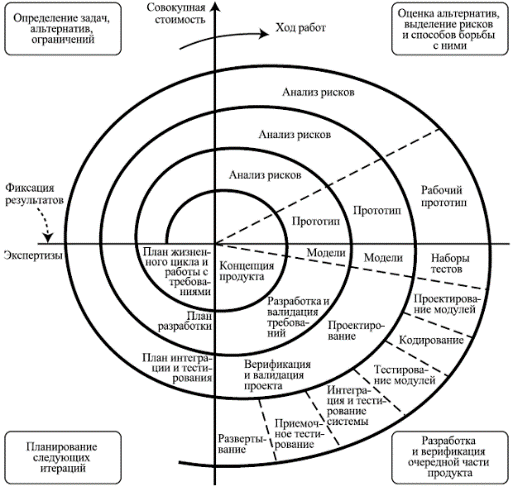
Существует множество моделей процессов жизненного цикла систем и программных средств, но три из них в международных стандартах обычно квалифицируются как фундаментальные: каскадная; инкрементная; эволюционная. ISO 12207

**Каскадная модель** реализует принцип однократного выполнения каждого из базовых процессов и этапов в их естественных границах.

Связь между этапами показана только сверху вниз, тогда как в реальных процессах ЖЦ следует учитывать возможность возврата на предшествующие этапы, снизу вверх, для их уточнения и корректировки результатов. При применении этой модели для создания каждого программного компонента соответствующие работы и задачи ЖЦ обычно выполняют последовательно. Однако они могут быть частично выполнены параллельно в случаях перекрытия последовательных работ.

**Инкрементная модель** ЖЦ подразумевает разработку информационной системы с линейной последовательностью стадий, но в несколько инкрементов (версий). Данная модель ЖЦ характерна при разработке сложных и комплексных систем, для которых имеется четкое видение того, что собой должен представлять конечный результат. Разработка версиями ведется в силу разного рода причин:

* отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;
* отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;
* требований поэтапного внедрения и освоения продукта конечными пользователями. Внедрение всей системы сразу может «затормозить» процесс перехода на новые технологии.

**Спиральная модель**. Отличием спиральной модели является особое внимание рискам, влияющим на организацию ЖЦ информационной системы. Каждый виток спирали соответствует поэтапной модели создания части системы, для которой уточняются цели и характеристики проекта. Существует проблема — определение момента перехода на следующий этап. Решение ее — ввод временных рамок, которые будут ограничивать время жизни каждого из этапов ЖЦ. Недостаток модели состоит в том, что ошибки могут быть допущены на этапах анализа и проектирования, это может привести к дополнительным затратам или неуспеху всего проекта в целом.

**Основные задачи современной программной инженерии**: управление проектами ПС и разработка средств, методов и теорий, необходимых для обеспечения ЖЦ комплексов программ.

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

Стандарт ISO 13210:1998 Методы тестирования для оценки соответствия стандартам POSIX

Методы и стратегии тестирования программных компонентов используются для обработки текстов программ в различной форме и для представления информации о результатах тестирования в виде удобном для анализа специалистами. Программы в процессе тестирования используются в двух принципиально различных формах представления:

- в виде текста на языке программирования или формализованного описания спецификаций требований (символьное представление), удобного для анализа человеком и обычно недоступного для непосредственного получения результатов функционирования на ЭВМ;

- в машинном коде конкретной ЭВМ (объектное представление), пригодном для автоматической обработки определенных кодовых исход-ных данных и неудобном для их анализа человеком.

**Тестирование потоков управления** (структуры программы) должно быть начальным этапом, так как при некорректной структуре возможны наиболее грубые искажения выходных результатов и даже отсутствие некоторых из них. Оно состоит в проверке корректности последовательностей передач управления и формирования маршрутов исполнения программы при обработке тестов. Для тестирования структуры программ в большинстве случаев требуются относительно меньшие затраты по сравнению с тестированием на потоках данных.

**Тестирование потоков данных** можно разделить на два этапа. Первый этап тестирования состоит в анализе обработки данных, определяющих значения предикатов в операторах выработки логических решений. Эти решения влияют на маршруты обработки информации, что сближает в этой части метод тестирования потоков данных с тестированием структуры программы. Второй этап тестирования обработки данных состоит в проверке вычислений по аналитическим формулам или численных значений результатов в зависимости от числовых или логических значений исходных данных. В качестве эталонов используются результаты ручных или автоматизированных расчетов по тем же или близким по содержанию формулам.

**Процессы и средства тестирования программных компонентов**

**Верификация** — процесс определения, выполняют ли программные средства и их компоненты требования, наложенные на них в последовательных этапах ЖЦ ПС. Важнейшей частью являются: анализ, обзоры и тестирование от требований. **Основная цель верификации** ПС - обнаружение, регистрация и устранение дефектов и ошибок, внесенных во время разработки и модификации программ. Верификация должна быть интегрирована как можно раньше с процессами проектирования, разработки и сопровождения. Обычно проводится сверху-вниз (от общих требований до детальных).

**Назначение верификации** ПС — последовательно проверить, что:

* ● общие требования к ИС, корректно переработаны в спецификацию требований высокого уровня к комплексу программ, удовлетворяющих исходным системным требованиям;
* ● в спецификации требований к функциональным компонентам низкого уровня, удовлетворяющих требованиям высокого уровня, требования высокого уровня правильно переработаны в архитектуру ПС;
* ● архитектура ПС и спецификации требований к компонентам низкого уровня корректно переработаны в исходные тексты программных и информационных модулей;
* ● исполняемый объектный код удовлетворяет требованиям к исходному тексту программных компонентов.

Верификации на соответствие спецификации требований на конкретный проект ПС также подлежат требования к технологическому обеспечению ЖЦ ПС и требования к эксплуатационной и технологической документации.

**Цели верификации** ПС достигаются посредством последовательного выполнения комбинации из просмотров, анализов, разработки тестовых сценариев и процедур и последующего выполнения этих процедур.

**Тестирование**: В качестве объекта тестирования выступают функциональные требования, требования к безопасности, удобству применения, взаимодействия компонент между собой и внешней средой и несколько особняком стоит тестирование, связанное с изменениями:

Далее рассмотрим классификацию тестирования по уровням:

* ● Компонентное (модульное) тестирование (Сomponent/Unit testing)
* ● Интеграционное тестирование (Iintegration testing)
* ● Системное тестирование (System/end-to-end testing)
* ● Приемочное тестирование (Acceptance Testing)

Модульное или компонентное тестирование направлено на поиск дефектов и ошибок в небольших частях ПО: модулях, функциях, классах и т.д.

Интеграционное тестирование предназначено для проверки взаимодействия между компонентами, ошибок в межмодульном интерфейсе, а также взаимодействия с операционной системой, аппаратурой, внешней средой.

Основной задачей системного тестирования является проверка всех функциональных, требований в системе в целом; оно может выявить непроверенные сочетания входных данных, применение ПО в системе для этого непредусмотренной, несогласованное или неадекватное использование ресурсов, отсутствующую или неверная функциональность, неудобство использования и т.д.

Приемочное тестирование определяет, удовлетворяет ли система приемочным критериям.

Достаточно часто используют терминологию, отражающую знание тестировщика о системе:

● Тестирование чёрного ящика (black box): тестировщик имеет доступ к ПО только через интерфейс; обычно критерий покрытия складывается из покрытия структуры входных данных и покрытия требований.

● Тестирование белого ящика (white box): разработчик теста имеет доступ к исходному коду; чаще всего применяется в случае модульного тестирования; используются метрики покрытия кода

● Тестирование серого ящика (grey box): разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется. Также тестирование может быть классифицировано по степени автоматизации:

● Ручное тестирование (manual testing): ручное выполнение тестов и проверка результатов;

● Автоматизированное тестирование (automated testing): использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс;

● Полуавтоматизированное тестирование (semiautomated testing): смесь.

Одной из главных проблем автоматизированного тестирования является его трудоемкость: большие ресурсы могут тратиться на обновление самих тестов. Как правило, автоматизированное тестирование является лишь дополнением ручного тестирования и не может его заменить полностью. Существует классификация по степени формализации процесса тестирования:

● Тестирование по документации (formal testing)

● Тестирование ad hoc или тестирование без определенного плана и документации, основанное на импровизированном подходе к поиску ошибок (adhoc testing)

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

**Проблемы современной программной инженерии:**

* разнородность ПС
* частое отсутствие аналогов, что не позволяет использовать типовые решения
* сложность определения требований в программной системе
* разобщенность и разнородность отдельных групп разработчиков
* трудность тестирования больших программных продуктов
* потребность в увеличении степени дублирования кода
* сложность перехода от индивидуального программирования, к коллективному
* сложность использования потенциальных возможностей аппаратуры компьютера
* сложность модернизации ранее существующих программных систем.

**Система называется сложной**, если в ней не хватает ресурсов (информационных) для эффективного описания (состояний, законов функционирования) и управления системой.

* **Внутренняя сложность** — у системы сложная внутренняя структура — много внутренних состояний, которые проявляются в сложности управления системой.
* **Внешняя сложность** — обусловлена сложностью взаимодействия с окружающей средой. Много контактов, затруднен учет влияний.
* **Большая система** — если её исследование, моделирование, разработка затруднено из-за большой размерности.

Сложности в сложных системах бывают:

* структурные или статические слишком много компонентов, не хватает ресурсов для построения, описания, управления структурой);
* динамические или временные (не хватает ресурсов для описания динамики поведения системы и управления ее траекторией);
* информационные или инфологические (не хватает ресурсов для информационного, информационно-логического описания системы);
* вычислительные или реализации, исследования (не хватает ресурсов для эффективного прогноза, расчетов параметров системы или их проведение затруднено нехваткой ресурсов);
* алгоритмические или конструктивные (не хватает ресурсов для описания алгоритма функционирования или управления системой, для функционального описания системы);
* развития или эволюции, самоорганизации (не хватает ресурсов для устойчивого развития, самоорганизации).

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

**Интеграционное тестирование** ‒ объединение программного кода, соответствующего двум или большему количеству программных модулей, и тестирование полученного в результате кода. Это должно гарантировать, что вместе они работают, как требуется. Подходы к интеграционному тестированию:

● Снизу-вверх. Все низкоуровневые модули собираются воедино → тестируются → собирается следующий уровень. Подход полезен, если все или практически все модули, разрабатываемого уровня, готовы. Помогает определить по рез-там тестирования уровень готовности приложения.

● Сверху-вниз. (модули низкого уровня симулируются заглушками).

● Большой взрыв. Собираются рабочие модули и тестируются. Экономия времени.

**Квалификационное тестирование**: Тестирование, выполняемое с целью убедить заказчика, что ПО соответствует заданным требованиям (ISO 12207).

**Испытания** компонентов и сложного комплекса программ должны включать:

● организацию и процессы испытаний компонентов и комплекса программ:

○ определение задач, процессов и методической достоверности испытаний компонентов и КП;

○ подготовку к интеграции компонентов, комплекса программ и аппаратуры системы;

● интеграцию и тестирование комплекса программ вне системы:

○ испытания интерфейсов, компонентов и КП на соответствие требованиям и эталонам;

○ оценка реализуемости и планирование испытаний комплекса программ в составе системы;

○ анализ полноты и корректности документации на комплекс программ;

● приемо-сдаточные испытания программного комплекса в составе системы:

○ испытания соответствия функций и характеристик качества программного продукта и системы требованиям и эталонам контракта;

○ удостоверение адекватности и качества технологической и эксплуатационной документации программного продукта требованиям на систему;

○ оформление акта о завершении работ и контракта на создание прогр. продукта и системы;

● опытную эксплуатацию – Альфа и Бета испытания в системе и пользователями;

● завершение испытаний на соответствие требованиям и утверждение готовности программного продукта для предъявления заказчику и поставки пользователям;

● внедрение версии программного продукта для применения в системе и обучение пользователей;

● изменения требований к КП, создание и сопровождение новых версий программного продукта.

Испытываться может лишь система, имеющая ТЗ. Сначала составляется общий план оценивания (сеть, ПО, оборудование, надёжность, безопасность). Затем проводятся сами испытания: проверяются функциональные характеристики (**реактивность** (способность выдерживать заданные интервалы между запуском и ответом), кол-во запросов, обрабатываемых в единицу времени и т.п.), **эргономические** характеристики интерфейса, **отказоустойчивость**.

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

**Профиль стандартов** — это совокупность нескольких базовых стандартов с четко определенными подмножествами обязательных и факультативных возможностей, предназначенная для реализации заданной функции или группы функций.

**Цели применения профилей стандартов при создании и применении ПС**:

* улучшение технико-экономических показателей проектов;
* повышение качества разрабатываемых/применяемых компонентов и ПС в целом при их разработке, приобретении, эксплуатации и сопровождении;
* обеспечение расширяемости и масштабируемости в зависимости от размерности решаемых задач;
* поддержка функциональной интеграции в системах задач, ранее решавшихся раздельно;
* обеспечение переносимости программ и данных между разными аппаратно-программными платформами.

Профиль стандартов ЖЦ ПС должен определять архитектуру программного комплекса и их структуру. ЖЦ ПС отражается в профиле стандартов набором процессов, этапов, частных работ и операций в последовательности их выполнения и взаимосвязи, регламентирующим ведение разработки, сопровождение и эксплуатацию, от анализа и подготовки требований до завершения испытаний ряда версий программного продукта и прекращения их использования.

Целесообразно рассматривать две группы профилей стандартов систем:

1. **функциональные** профили, регламентирующие архитектуру и структуру объектов системы и ее компонентов; функции, интерфейсы и протоколы взаимодействия, форматы данных;

2. **технологические** профили, регламентирующие процессы проектирования, разработки, применения, сопровождения и развития систем и их компонентов.

На этапах ЖЦ системы применяются **общесистемные функциональные** профили:

1. профиль ЖЦ информационной системы;
2. профиль аппаратной и операционной среды системы;
3. профиль внешней и пользовательской среды функционирования ПС;
4. профиль обеспечения безопасности функционирования и защиты информации в системе;
5. профиль инструментальных средств, поддерживающих весь ЖЦ системы.

Применение функциональных профилей должны поддерживать **основные, технологические профили**:

1. ЖЦ ПС и БД;
2. обеспечения качества ПС и информации БД;
3. верификации, тестирования и сертификации ПС и БД;
4. сопровождения и управления конфигурацией ПС и информацией БД;
5. документирования ПС и информации БД.

По мере детализации структуры системы и ее возможного развития образует ЖЦ профилей стандартов. **ЖЦ профилей ПС** целесообразно рассматривать в составе технологических работ проекта отдельно от этапов и работ непосредственной разработки и эксплуатации самих ПС и БД. Создание и применение профилей ЖЦ ПС можно разделить на два процесса:

* разработка, формирование и адаптация профилей стандартов ЖЦ ПС для использования в конкретном проекте системы;
* непосредственное применение требований и рекомендаций каждого адаптированного профиля
* стандартов для регламентирования этапов, работ и документов проекта ПС.

При применении профилей следует обеспечить проверку корректности их использования путем тестирования, испытаний и сертификации. Профили должны определяться таким образом, чтобы тестирование их реализации можно было осуществлять по возможности наиболее полно, по стандартизированной методике.

Стандарт **ISO/IEC 12207** (ИТ. Системная и программная инженерия. Процессы ЖЦ ПС) не предлагает конкретную модель ЖЦ и методы разработки ПО. Его регламенты являются общими для любых моделей ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

В соответствии с базовым международным стандартом ISO/IEC 12207 **все процессы ЖЦ ПО делятся на три группы**:

1. Основные процессы: приобретение; поставка; разработка; эксплуатация; сопровождение.
2. Вспомогательные процессы: документирование; управление конфигурацией; обеспечение качества; разрешение проблем; аудит; аттестация; совместная оценка; верификация.
3. Организационные процессы: создание инфраструктуры; управление; обучение; усовершенствование.

В стандарте **ISO 12207** и Приложениях 1 и 2 к этому стандарту, изложены основы преобразования и адаптации базовой структуры процессов ЖЦ для профиля конкретного проекта ПС и БД. В них даны общие рекомендации по адаптации процессов ЖЦ, а также рекомендации по возможным изменениям ряда работ и результирующих документов.

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.

**Процесс эксплуатации ПС** ‒ использование ПС по его непосредственному предназначению.

Состоит из работ и задач оператора. Процесс охватывает эксплуатацию программного продукта и поддержку пользователей в процессе эксплуатации. Работы (процедуры), обеспечивающие эксплуатацию ПС, в соответствии с требованиями ISO 12207 (ИТ. Системная и программная инженерия. Процессы ЖЦ ПС):

1. подготовка процесса: оператор разрабатывает план эксплуатации, устанавливает процедуры получения сведений о возникающих проблемах и их решения;
2. эксплуатационные испытания (эксплуатационное тестирование): тестирование ПС в ее рабочем окружении, также обеспечение установки начального состояния компонентов системы (н-р, БД);
3. непосредственно эксплуатация системы в установленной эксплуатационной среде в соответствии с требованиями;
4. поддержка пользователя: помощь и консультации в установленном порядке; при необходимости запросы о поддержке направляются в процесс сопровождения (для внесения исправлений в ПС).

**Процесс сопровождения ПС** ‒ выявление и устранение обнаруженных дефектов и ошибок, введение новых функций и компонентов в ПС, анализ состояния и корректировка документации, тиражирование и контроль распространения версий ПС, актуализация и обеспечение сохранности документации и физических носителей. **Основная задача** — изменить и улучшить существующий программный продукт, сохраняя его целостность и функциональную пригодность. В соответствии с требованиями ISO 12207 по развитию и модификации программного продукта в ЖЦ должен быть организован процесс его сопровождения. Работы (процедуры), обеспечивающие сопровождение ПС, включают:

* подготовку процесса;
* анализ проблем и изменений;
* внесение изменений;
* проверку и приемку при сопровождении;
* перенос;
* снятие с эксплуатации.

При подготовке процесса сопроводитель должен создать планы и определить процедуры, выполняемые при реализации сопровождения (см список ниже). План сопровождения целесообразно создавать параллельно с планом разработки первой, базовой версии ПС. При выполнении данной работы сопроводитель также должен определить необходимые организационные интерфейсы и взаимосвязи между специалистами и с другими предприятиями. Исходными данными для работ по подготовке процесса являются: старая (исходная) базовая версия программного продукта; системные документы; предложения о модификациях и отчеты о дефектах. **Целью планирования сопровождения** является подготовка плана работ по сопровождению и обеспечение ресурсов, необходимых для проведения этих работ после передачи программного продукта на сопровождение.

**Проектирование архитектуры модификаций** определяет функции и компоненты модифицированного программного средства. Основными особенностями данной работы среди процессов ЖЦ ПС, влияющими на сопровождаемость, являются выбор структуры программы, разбиение ее на компоненты (модули) и поток данных, циркулирующий между ними. При модификациях важно использовать знания коллектива специалистов по обработке данных, относящиеся к возможности использования частей существующих программ или библиотек, доказавших высокое функциональное качество.

**При внесении изменений в ПС** сопроводитель разрабатывает и тестирует конкретные изменения программного продукта. Исходными данными для проведения работы при внесении изменений должны быть: базовая версия программного продукта; согласованные с заказчиком предложения о модификации; согласованные документы на реализацию корректировки; отчет о влиянии корректировки и выходные результаты работы по анализу изменений.

Для плавного перехода к новой базовой версии (**переноса)** программного продукта должна быть обеспечена параллельная эксплуатация прежнего и нового программных продуктов.

**Снятие программного средства с эксплуатации** и сопровождения должно быть подготовлено анализом, обосновывающим это решение. В анализе следует определить и экономически обосновать: возможность сохранения устаревшей версии комплекса программ, а также необходимость создания и применения новой версии программного продукта.

Пользователи должны получать уведомления о любых готовящихся работах по переносу или снятию с эксплуатации.

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.

**Методология обеспечения качества ПС** в программной инженерии поддержана рядом методических документов и инструментальных средств, а также формализована комплексом международных стандартов. Внедрение комплекса требует больших усилий и затрат, что ограничило его массовое использование для относительно простых и средней сложности проектов. Концептуальные и организационные основы административного управления ЖЦ и качеством ПС в системе СММ определены в восьми базовых принципах, которые декларированы в стандартах ISO 9000 (Системы менеджмента качества) и ISO 15504 (ИТ. Оценка процессов).

1. Ориентация предприятия-разработчика на потребителя-заказчика.
2. Лидерство-руководство.
3. Вовлечение персонала.
4. Процессный подход. (управлять ресурсами и деятельностью как целым).
5. Системный подход к административному управлению. (административное управление системой взаимосвязанных процессов для заданной стратегической цели повышает эффективность и результативность предприятия)
6. Постоянное усовершенствование.
7. Подход к принятию решений, основанный на фактах.
8. Взаимовыгодные отношения с поставщиками.

В стандарте **ISO 15504** каждый из приведенных принципов прокомментирован комплексом действий, необходимых для их реализации в проектах. Эти принципы рекомендуется применять при:

1. формулировке политики и стратегии обеспечения всего ЖЦ ПС;
2. выборе целей проекта, требований и характеристик качества ПС, непосредственно связанных с потребностями и ожиданиями заказчиков и потребителей;
3. управлении операциями в процессе реализации проекта и для удовлетворения требований заказчика и потребителей;
4. управлении людскими ресурсами предприятия для обеспечения ЖЦ ПС и его качества.

Описание процессов ЖЦ ПС в **СММ** (Capability Maturity Model) сфокусировано на поэтапном определении реально достигаемых результатов и на оценивании качества их выполнения. Качество процессов зависит от технологической среды, в которой они выполняются.

В современных автоматизированных технологиях программной инженерии, создания и совершенствования сложных ПС с позиции обеспечения их качества можно выделить методы и средства, позволяющие:

1. создавать программные модули и функциональные компоненты высокого, гарантированного качества;
2. предотвращать дефекты проектирования за счет систем обеспечения качества, эффективных технологий и инструментальных средств автоматизации всего ЖЦ комплексов программ и БД;
3. обнаруживать и устранять различные дефекты и ошибки проектирования, разработки и сопровождения программ путем верификации и систематического тестирования на всех этапах ЖЦ ПС;
4. удостоверять достигнутые значения качества функционирования программных продуктов в процессе их испытаний и сертификации перед передачей в регулярную эксплуатацию пользователям.

Комплексное, скоординированное применение этих методов и средств в процессе создания, развития и применения ПС позволяет исключать многие виды дефектов или значительно ослаблять их влияние.

Для обнаружения, устранения ошибок и дефектов все этапы разработки и сопровождения ПС должны быть поддержаны методами и средствами верификации и тестирования корректности реализованных решений.

При применении импортных компонентов в системном проектировании и обеспечении качества программных продуктов следует учитывать, что, в принципе, в них возможны как злоумышленные, так и случайные, непредумышленные дефекты вычислительного процесса, программ и данных, отражающиеся на качестве их функционирования.

Для удостоверения качества, надежности и безопасности применения сложных, критических систем используемые в них программные продукты следует подвергать сертификации аттестованными, проблемно-ориентированными испытательными центрами. Необходимо, когда дефекты или недостаточное качество могут нанести значительный ущерб.

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

Процесс УК направлены на достижение следующих **основных целей**:

● обеспечить возможность оценки соответствия требованиям заказчика результатов ЖЦ ПС;

● обеспечить определяемую и 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 эл-тами конфигурации и определения базовой версии прогр. продукта;

● обеспечить контроль над тем, чтобы фиксировались дефекты и ошибки, а изменения регистрировались, утверждались и реализовывались;

● гарантировать надежное физическое архивирование, восстановление и сопровождение единиц конфигурации и документов программного продукта.

По ISO 12207 данный процесс состоит из следующих работ:

1. Подготовка процесса. Задача: разработка плана управления конфигурацией.
2. Определение конфигурации. Задача: определение схемы обозначения ПС и ее версий.
3. Учет состояний конфигурации. Задача: обозначение и регистрация заявок на внесение изменений в ПС и их выполнение или отклонение.
4. Оценка конфигурации. Задача: определение и обеспечение функциональной законченности и физической завершенности ПС.
5. Управление выпуском и поставка. Задача: контроль выпуска и поставки ПС с соответствующей документацией.

**План управления конфигурацией** должен определять: работы по управлению конфигурацией; процедуры и график выполнения данных работ; организацию(и), ответственную(ые) за выполнение данных работ; связь данной организации(й) с другими организациями, например, по разработке и сопровождению программных средств. План должен быть документально оформлен и выполнен.

Цель **отчетности о состоянии конфигурации** состоит в обеспечении информации для УК процессов ЖЦ ПО относительно идентификации конфигурации, базовых линий, сообщений о дефектах и контроля изменений. Работы, связанные с отчетностью о состоянии конфигурации:

● регистрация идентификации элементов конфигурации, идентификации базовых линий, состояния сообщений о дефектах, хронологии изменений и состояния выпускаемой линии;

● определение информации, подлежащей сопровождению, и способов регистрации и ведения отчетности о состоянии конфигурации этой информации.

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

Стандарт ISO 9126:1-4 (ИТ. Оценка программной продукции. Характеристики качества и руководства по их применению) целесообразно использовать как основу для формального регламентирования характеристик качества в ЖЦ проектов ПС. Модель характеристик качества ПС и компонентов состоит из 6 групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками:

● Функциональные возможности детализируются:

○ пригодностью для применения по назначению;

○ корректностью (правильностью, точностью) реализации требований;

○ способностью к взаимодействию с компонентами и средой;

○ защищенностью – безопасностью функционирования.

● Надежность характеризуется:

○ уровнем завершенности – отсутствием дефектов и ошибок;

○ устойчивостью при наличии дефектов и ошибок;

○ восстанавливаемостью после проявления дефектов;

○ доступностью – готовностью реализации требуемых функций.

● Эффективность рекомендуется отражать: временной эффективностью реализации комплекса программ; используемостью вычислительных ресурсов.

● Применимость (практичность) предлагается описывать:

○ понятностью функций и документации;

○ простотой использования комплекса программ;

○ изучаемостью процессов функционирования и применения.

● Сопровождаемость представляется: анализируемостью – удобством для анализа предложений модификаций; изменяемостью компонентов и комплекса программ; тестируемостью изменений при сопровождении.

● Мобильность (переносимость) предлагается отражать:

○ адаптируемостью к изменениям среды;

○ простотой установки – инсталляции после переноса;

○ замещаемостью компонентов при корректировках комплекса программ.

Характеристики и субхарактеристики в стандарте определены кратко, без комментариев и подробных рекомендаций по их применению к конкретным системам и проектам ПС. Изложение имеет концептуальный характер и не содержит рекомендаций по выбору и упорядочению приоритетов, а также необходимого минимума критериев в зависимости от особенностей объекта, среды разработки, сопровождения и применения.

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.

**Цель сертификации ПС** - контроль и удостоверение качества технологий и продукции, гарантирование их высоких потребительских свойств. Задача - повышение эффективности затрат в сфере создания и применения конечного программного продукта и улучшение объективности оценок его характеристик и конкурентоспособности. **Формальная цель сертификации** - подготовка и принятие решения о целесообразности выдачи заявителю сертификата соответствия **с учетом следующих факторов**:

* полноты, точности и достоверности исходного ТЗ и спецификаций требований, представленных в документации на ПС, а также на технологию поддержки его ЖЦ;
* достоверности и точности измерения и обобщения результатов сертификационных испытаний и получения адекватных показателей качества конечных программных продуктов и/или технологических процессов их создания;
* методологии и качества интерпретации данных об объекте испытаний и/или технологии с учетом достоверности оценок, квалификации и объективности испытателей, заказчиков и пользователей.

**Результатом положительных испытаний является сертификат соответствия** - документ, изданный по правилам Системы сертификации, удостоверяющий, что обеспечивается необходимая уверенность в том, что должным образом идентифицированная продукция, процесс или услуга соответствует конкретным стандартам и/или другим нормативным документам. Срок действия сертификата обычно ограничен либо по времени (например, 3 года), либо до проведения достаточно значительной модификации продукта или процесса. Сертификат вступает в действие с момента его регистрации в государственном реестре.

**Базовые документы системы качества предприятия и ЖЦ программного средства**

1. Концепция, терминология, требования и Руководство по улучшению деятельности – **ISO 9000:2000** (Системы менеджмента качества) или версия модели зрелости СММI.
2. Адаптированные версии или перечень разделов и рекомендаций стандартов **ISO 12207** (ИТ. Системная и программная инженерия. Процессы ЖЦ ПС), **ISO 15504** (ИТ. Оценка процессов), их Изменений и Руководств по применению, выделенных при адаптации и обязательных для использования в системе качества конкретного предприятия или проекта программного продукта.
3. Адаптированная версия стандарта ISO 90003 (Руководящие указания по применению **ИСО 9001:2008** (Системы менеджмента качества. Требования) при разработке программных продуктов), выделенных при адаптации и обязательных для применения в системе качества предприятия, выпускающего программный продукт.
4. Базовые характеристики и атрибуты качества проекта ПС, выделенные, адаптированные и конкретизированные на основе стандартов **ISO 12182** (ИТ. Классификация ПС), **ISO 9126** (ИТ. Оценка программной продукции. Характеристики качества и руководства по их применению), **ISO 14598** (Метод оценки качества ПС)
5. Адаптированная версия и утвержденная редакция Руководства по сопровождению и конфигурационному управлению основе рекомендаций стандартов **ISO 14764** (ИТ. Сопровождение ПС), **ISO 10007** (Менеджмент организации. Руководящие указания по управлению конфигурацией), **ISO 15846** (ИТ. Процессы ЖЦ ПО. Управление конфигурацией).
6. Комплект должностных инструкций, определяющих ответственность, полномочия и порядок взаимодействия всего руководящего, выполняющего и проверяющего работу персонала, участвующего в процедурах системы качества предприятия для конкретного проекта ПС.

**Исходные документы, отражающие особенности ЖЦ конкретного программного средства:**

1. **Описание характеристик программных продуктов**, создаваемых на предприятии, системы и внешней среды их ЖЦ, необходимых для адаптации и подготовки рабочих версий стандартов и требований проекта ПС и системы качества предприятия в соответствии с рекомендациями стандартов ISO 12207, ISO 15504, ISO 90003 и ISO 9126.
2. **Описание целей, требований и обязательств предприятия-разработчика** в области системы качества, критериев качества процессов и продуктов разработки, поставки и поддержки всего ЖЦ ПС.
3. **Комплект эксплуатационных документов, поставляемых заказчику** и пользователям для обеспечения ЖЦ и применения конкретной версии программного продукта на основе адаптированных стандартов ISO 9294 (ИТ. Рук-во по управлению документированием ПО), ISO 15910 (ИТ. Процесс создания документации пользователя ПС), ISO 18019 (ПО и системотехника. Рекомендации по проектированию и подготовке документации пользователя по прикладному
4. ПО).
5. **Документация и средства автоматизации** проектирования, разработки, модификации, контроля и испытаний, используемых для обеспечения ЖЦ программного продукта.
6. **Планы и методики испытаний** применения и оценки эффективности процессов системы качества предприятия и программного продукта.
7. **Методики сопровождения**, идентификации компонентов программного продукта и документации, анализа и утверждения версий комплексов программ и данных.
8. **Методика конфигурационного управления**, утверждения, хранения, защиты, копирования версий программного продукта и сопровождающих документов, а также накопления и хранения, зарегистрированных в архиве предприятия данных о характеристиках качества в течение ЖЦ версий программного продукта.
9. Управление проектами программных средств в системе СMMI

**Назначение методологии СММ/CMMI** (системы и модели оценки зрелости) 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. В методологии СММ выделены пять уровней зрелости. Виды деятельности для высоких уровней зрелости в соответствии с СММ, в стандарте делятся на базовые и общие. Базовые виды деятельности являются обязательными и сгруппированы в пять категорий: контрактная; инженерная; управленческая; вспомогательная; организационная. **Уровни зрелости** характеризуются степенью формализации, адекватностью измерения и документирования процессов и продуктов ЖЦ ПС, широтой применения стандартов и инструментальных средств автоматизации работ, наличием и полнотой реализации функций системой обеспечения качества технологических процессов и их результатов.

**Описание процессов ЖЦ ПС в СММ** сфокусировано на поэтапном определении, реально достигаемых результатов и на оценивании качества их выполнения. Качество процессов зависит от технологической среды, в которой они выполняются. Зрелость процессов - это степень их управляемости, возможность поэтапной количественной оценки качества, контролируемости и эффективности результатов. Модель зрелости предприятия - методический нормативный материал, определяющий правила создания и функционирования системы управления ЖЦ ПС, методы и стандарты систематического повышения культуры и качества производства. Рост зрелости обеспечивает потенциальную возможность возрастания эффективности и согласованности использования процессов создания, сопровождения и оценивания качества компонентов и ПС в целом. Реальное использование регламентированных процессов предполагает поэтапный контроль и документирование их характеристик качества. На предприятиях, достигших высокого уровня зрелости, формализованные процессы ЖЦ ПС должны принимать статус стандарта, фиксироваться в организационных структурах и определять производственную тактику и стратегию корпоративной культуры производства и системы обеспечения качества ПС.

**Уровни зрелости по CMM**:

* Уровень 1 - Начальный.
* Уровень 2 - Управляемый – базовое управление.
* Уровень 3 - Определенный – стандартизация процессов.
* Уровень 4 - Предсказуемый – количественное управление
* Уровень 5 - Оптимизационный – непрерывное совершенствование и улучшение.

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

**Оценивание надежности ПС** включает измерение количественных суб-характеристик и их атрибутов: завершенности, устойчивости к дефектам, восстанавливаемости и доступности-готовности. Измерения проводятся при функционировании готового программного продукта для сопоставления с заданными требованиями и для оценивания степени соответствия этим спецификациями требований.

● Прямые экспериментальные методы оценивания интегральных характеристик надежности ПС трудно реализовать из-за больших значений времени наработки на отказ (сотни и тысячи часов), которые необходимо достигать при разработке и фиксировать при испытаниях. Сложность выявления и регистрации редких отказов и высокая стоимость экспериментов приводят к малому количеству зарегистрированных отказов => недостоверная оценка надежности.

● Для выявления тенденции изменения показателей надежности, их зарегистрированные значения необходимо связывать во времени с моментами корректировки программ и данных.

● При заключительных приемо-сдаточных и сертификационных испытаниях для достоверного определения надежности ПС организуются многочасовые и многосуточные прогоны функционирования комплекса программ в реальной и/или имитированной внешней среде в условиях широкого варьирования исходных данных с акцентом на стрессовые ситуации, стимулирующие проявления угроз надежности.

Форсированные испытания для оценивания надежности ПС осуществляется повышением интенсивности искажений исходных данных и расширением варьирования их значений, а также специальным увеличением интенсивности потоков информации и загрузки программ на ЭВМ выше нормальной.

При форсированных испытаниях целесообразно выделять следующие режимы тестирования:

● полное искажение, предельные и критические значения ключевых параметров каждого типа внешней информации и воздействий пользователей;

● предельные и критические сочетания значений различных взаимодействующих параметров эксплуатации ПС;

● предельно большие и малые интенсивности суммарного потока и каждого типа внешней информации;

● умышленные нарушения пользователями определенных положений инструкций и рекомендаций эксплуатационной документации.

Виды форсированного тестирования: бета-тестирование и тестирование эффективности средств оперативного контроля и восстановления программ, данных и вычислительного процесса. (Бета-тестирование - контроль результатов функционирования одних и тех же ПС при увеличении числа испытываемых экземпляров и нормальных исходных данных

**Оценивание ресурсной эффективности** состоит в измерении количественных субхарактеристик и их атрибутов: временной эффективности и используемости ресурсов ЭВМ комплексом программ (см. таблицу 11.3 вопрос 25). При этом предполагается, что в техническом задании и спецификации требований зафиксированы требуемые значения атрибутов и их приоритеты.

В стандарте ISO 9126:2 (ИТ. Оценка программной продукции. Характеристики качества и руководства по их применению) эту характеристику качества ПС рекомендуется отражать десятком атрибутов, каждый из которых оценивать для средних и наихудших сценариев функционирования комплекса программ. В таблице 11.3 сохранены только три атрибута, важнейшие для функционирования ПС и наиболее доступны количественным измерениям. Оценивание этих атрибутов может проводиться при функционировании готового программного продукта или расчетными методами, при разработке для сопоставления с заданными требованиями.

Для измерения атрибутов временной эффективности необходимы инструментальные средства, встроенные в операционную систему или в соответствующее ПС. Эти средства должны в динамике реального функционирования программ регистрировать: загрузку вычислительной системы; значения интенсивности потоков данных от внешних абонентов; длительность исполнения заданий; характеристики функционирования устройств ввода/вывода; время ожидания результатов (отклика) на задания пользователей; заполнение памяти обмена с внешними абонентами в различных режимах применения комплекса программ. Значения этих характеристик зависят не только от свойств и функций ПС, но также от особенностей архитектуры и операционной системы ЭВМ. Регулярная регистрация и обобщение таких данных позволяет выявлять ситуации, негативно влияющие на функциональную пригодность, надежность и другие конструктивные характеристики качества ПС.

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

В стандарте ISO 14252:1996 - Руководство по POSIX окружению открытых систем (OSE) – изложена идеология и модель создания мобильных ПС, которое детализирует для пользователей модель комплекса стандартов POSIX, а также взаимодействующих с ними стандартов де-юре, де-факто и спецификаций, необходимых для создания переносимых приложений. Модель отражает принципы построения интерфейсов прикладных программ с платформой - операционной системой, через которую осуществляется взаимодействие с компонентами внешнего окружения. Считается, что прикладные программы непосредственно не взаимодействуют с внешним окружением, а связаны с ним только через операционную систему. Разработка приложений предполагается в кросс-режиме, то есть платформа разработки – инструментальная может не совпадать с платформой исполнения (применения) программ (объектной – целевой). Результат компиляции программы на инструментальной платформе может быть перенесен для исполнения на целевую платформу.

Определяющими являются два интерфейса между тремя базовыми компонентами: между прикладными программами и платформой - операционной системой (API) и между платформой и внешним окружением (EEI). Определены общие функции - услуги платформы для этих взаимодействий. Внешнее окружение включает компоненты человеко-машинного взаимодействия с пользователями, компоненты информационного взаимодействия с внешними устройствами ЭВМ и с компонентами, обеспечивающими коммуникацию с внешней средой. Эти интерфейсы и услуги более детально формализованы соответствующими стандартами POSIX.

Стандарт включает также общие принципы и руководство по формированию и описанию сложных согласованных профилей и по обеспечению непротиворечивости их интерфейсов с внешним окружением:

* поддерживающие непосредственное взаимодействие прикладных программ с пользователями, регламентирующие интерфейсы с виртуальными терминалами и графические системы;
* обеспечивающие административное управление файловыми системами и различными, в том числе, распределенными базами данных;
* регламентирующие телекоммуникацию и обмен данными на верхних уровнях эталонной модели ВОС;
* обеспечивающие аттестацию и безопасность применения информационных технологий, программ и данных в соответствии со стандартами ВОС и криптографии.

**Стандарты открытых систем POSIX** регламентируют совокупность базовых, системных сервисов для обеспечения унифицированных интерфейсов прикладных программ, специфицированных для языка С, командного языка и совокупности служебных программ. Основная цель – сделать программы переносимыми на уровне различных исходных языков. У каждого интерфейса программ существует вызывающая и вызываемая сторона, стандарты POSIX ориентированы преимущественно на формализацию вызывающей стороны. Мобильность приложений должна обеспечиваться благодаря применению большого числа стандартизированных системных интерфейсных сервисов и возможности динамического выяснения характеристик целевой платформы и подстройки под них интерфейсов приложений.

Концептуально этот стандарт описывает набор фундаментальных услуг, необходимых для эффективной разработки прикладных программ. Доступ к этим услугам обеспечивался определением интерфейса, с использованием языка программирования Си, процессора командного языка, и стандартных сервисных программ, которые устанавливают стандартную семантику и синтаксис интерфейсов.

**Задачи POSIX:**

1. содействовать облегчению и автоматизации переноса кода готовых прикладных программ на иные платформы;
2. способствовать определению и унификации интерфейсов программных компонентов заранее при проектировании ПС, а не только в процессе их реализации;
3. сохранять по возможности и учитывать все главные, созданные ранее, унаследованные и используемые программные средства и компоненты;
4. определять необходимый минимум интерфейсов компонентов и комплексов программ, для ускорения создания и расширения программных продуктов, а также для анализа, одобрения и утверждения документов;
5. развивать стандарты в направлении обеспечения коммуникационных сетей, распределенной обработки данных и защиты информации;
6. рекомендовать ограничивать использование объектного кода для программ в простых системах.

По определению, принятому Комитетом IEEE POSIX 1003.0 (ISO 14252 Руководство по функциональной среде открытых систем POSIX) (набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API)) **открытой информационной системой** называется система, которая реализует открытые спецификации на интерфейсы, сервисы (услуги среды) и поддерживаемые форматы данных, достаточные для того, чтобы дать возможность должным образом разработанному прикладному ПО быть переносимым в широком диапазоне систем с минимальными изменениями, взаимодействовать с другими приложениями на локальных и удаленных системах, и взаимодействовать с пользователями в стиле, который облегчает переход пользователей от системы к системе.

**Свойства открытых систем**

1. Расширяемость;
2. Масштабируемость;
3. Переносимость приложений, данных и персонала;
4. Интероперабельность приложений и систем;
5. Способность к интеграции;
6. Высокая готовность.

**Открытой** может быть названа любая система (компьютер, вычислительная сеть, ОС, программный пакет, другие аппаратные и программные продукты), которая построена в соответствии с открытыми спецификациями.

**Новую версию POSIX** составляют четыре части стандартов ISO 9945:1-4:2003 (ИТ. Интерфейсы переносимых операционных систем).

1. Базовые определения.
2. Системные интерфейсы.
3. Команды управления и сервисные программы.
4. Обоснование.
5. Основные факторы, определяющие качество сложных программных средств. Метрики характеристик качества, описанные в стандарте ISO 9126:1-4:2002.

Метрики общего представления о качестве:

1. внутреннее качество, проявляющееся в процессе разработки и других промежуточных этапов ЖЦ ПС;
2. внешнее качество, заданное требованиями заказчика в спецификациях и определяется характеристиками конечного проекта;
3. качество при использовании в процессе нормальной эксплуатации и результативностью достижения потребностей пользователей с учетом затрат ресурсов.

Применение метрик:

Внутренние метрики – в ходе проектирования и программирования компонентов ПС, цель – обеспечить достижение внешнего качества.

Внешние метрики – в ходе испытаний, эксплуатации и наблюдения исполняемых программ, цель – обеспечить заказчикам, пользователям и разработчикам возможность анализировать качество ПС в ходе эксплуатации.

Метрики качества в использовании – при оценке удовлетворения потребностей пользователей, цель – интегральная оценка результатов.

Стандарт **ISO 9126:1-4** − целесообразно использовать как основу для формального ***регламентирования характеристик качества в жизненном цикле проектов программных средств***. Модель характеристик качества ПС и компонентов состоит из шести групп базовых показателей, каждая из которых детализирована несколькими нормативными субхарактеристиками:

***Функциональные возможности*** детализируются:

* пригодностью для применения по назначению;
* корректностью (правильностью, точностью) реализации требований;
* способностью к взаимо­действию с компонентами и средой;
* защищенностью – безопасностью функционирования.

***Надежность*** характеризуется:

* уровнем завер­шенности – отсутствием дефектов и ошибок;
* устойчивостью при наличии дефектов и ошибок;
* восстанавливаемостью после проявления дефектов;
* доступностью – готовностью реализации требуемых функций.

***Эффективность*** рекомендуется отражать:

* временной эффективностью реализации комплекса программ;
* используемостью вычислительных ресурсов.

***Применимость (практичность)*** предлагается описывать:

* понятностью функций и документации;
* простотой использования комплекса программ;
* изучаемостью процессов функционирования и применения.

***Сопровождаемость*** представляется:

* анализируемостью – удобством для анализа предложений модификаций;
* изменяемостью компонентов и комплекса программ;
* тестируемостью изменений при сопровождении.

***Мобильность (переносимость)*** предлагается отражать:

* адаптируемостью к изменениям среды;
* простотой установки – инсталляции после переноса;
* замещаемостью компонентов при корректировках комплекса программ.

**Метрики характеристик качества (ГОСТ Р ИСО/МЭК 9126-93)**

● Функциональные возможности (Functionality)

● Пригодность (Suitability):

○ Правильность (Accuracy)

○ Способность к взаимодействию (Interoperability)

○ Согласованность (Compliance)

○ Защищенность (Security)

● Надежность (Reliability):

○ Стабильность (Maturity)

○ Устойчивость к ошибке (Fault tolerance)

○ Восстанавливаемость (Recoverability)

● Практичность (Usability):

○ Понятность (Understandability)

○ Обучаемость (Learnability)

○ Простота использования (Operability)

● Эффективность (Efficiences):

○ Характер изменения во времени (Time behavior)

○ Характер изменения ресурсов (Resource behavior)

● Сопровождаемость:

○ Анализируемость:

■ Стройность архитектуры программы

■ Унифицируемость интерфейса

■ Полнота и корректность

○ Тестируемость: (чел/час)

■ Трудоемкость тестирования изменений

■ Длительность подготовки изменений.

● Мобильность:

○ Адаптируемость

■ Трудоемкость адаптации

■ Длительность адаптации.

○ Простота установки

■ Трудоемкость установки

■ Длительность установки

○ Замещаемость

■ Трудоемкость замены компонентов (час/час)

■ Длительность замены компонентов (час)

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

Система называется **сложной,** если в ней не хватает ресурсов (информационных) для эффективного описания (состояний, законов функционирования) и управления системой.

**Внутренняя сложность** — у системы сложная внутренняя структура — много внутренних состояний, которые проявляются в сложности управления системой.

**Внешняя сложность** — обусловлена сложностью взаимодействия с окружающей средой. Много контактов, затруднен учет влияний.

**Большая** — исследование, моделирование, разработка затруднена из-за большой размерности.

**Сложности в сложных системах бывают**:

1. структурные или статические (слишком много компонентов системы, не хватает ресурсов для построения, описания, управления структурой, приходится упрощать);
2. динамические или временные (не хватает ресурсов для описания динамики поведения системы и управления ее траекторией);
3. информационные или инфологические (не хватает ресурсов для информационного, информационно-логического описания системы);
4. вычислительные или реализации, исследования (не хватает ресурсов для эффективного прогноза, расчетов параметров системы или их проведение затруднено нехваткой ресурсов);
5. алгоритмические или конструктивные (не хватает ресурсов для описания алгоритма функционирования или управления системой, для функционального описания системы);
6. развития или эволюции, самоорганизации (не хватает ресурсов для устойчивого развития, самоорганизации).

**Технология проектирования** информационных систем состоит из **трёх важных составляющих:**

1. пошаговая процедура, определяющая технологическую последовательность шагов проектирования
2. критерии и правила оценки полученных на каждом технологическом этапе результатов
3. выбор нотации, графической и текстовой, для описания всей проектируемой системы Системное проектирование ПС

**Проект** - комплекс формально организованных мероприятий по достижению единой цели для создания сложной информационной системы с требуемыми характеристиками качества при ограниченных ресурсах.

Методологическая база целевого проекта - **системный анализ. Основная цель системного проектирования** - подготовить, обосновать и согласовать замыслы и решения между заказчиком и потребителем и учесть все требования к качеству и ограниченные ресурсы; в программной инженерии - подготовить, обосновать и согласовать запросы заказчика и разработчика. В ходе системного проектирования производятся: обзор предметной области, анализ рынка, анализ существующих

продуктов, выбор из списка альтернатив по методикам разработки (критерии - эффективность,

стоимость), учёт возможных рисков.

В ходе проектирования производится декомпозиция крупного проекта на более мелкие подсистемы

и построение их в виде иерархического дерева. Она производится таким образом, чтобы

минимизировать связи между подсистемами.

**Что нужно учесть на этапе системного проектирования:**

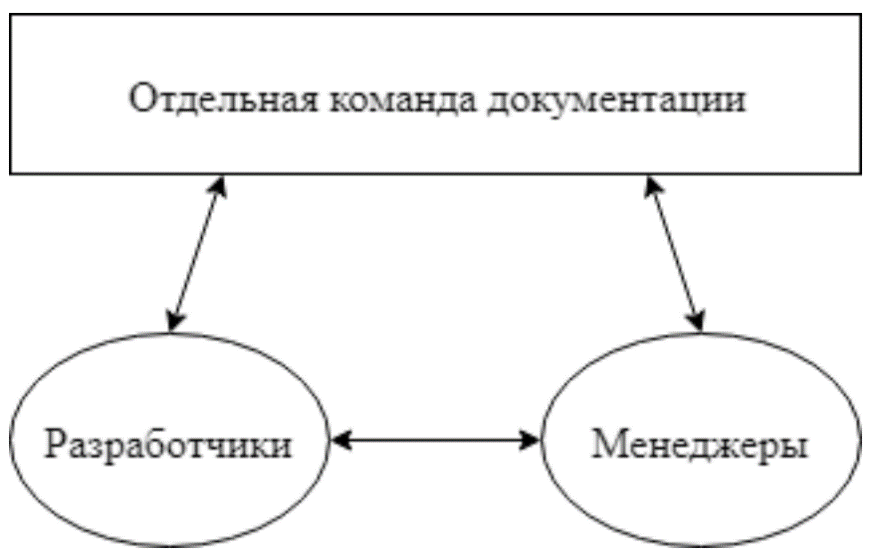
1. Цена ошибки минимальна на этапе системного проектирования. Поэтому стоит пригласить эксперта для оценки структуры проекта, обсудить структуру с пользователями и заказчиками.
2. В системном проекте прописать вопросы интероперабельности: как ваша система должна взаимодействовать с существующими системами заказчика.
3. Выделить общий, единый критерий качества для всей системы, глобальный для всех её подсистем. Выбор единого критерия является сложной, но необходимой задачей.
4. Выделение информационного объекта (информация на узлах системы), определение протоколов
5. Определить, будет ли вестись журналирование и сбор данных, каких данных и каким образом.

**Процессы системного проектирования ПС:**

1. Обследование объекта информатизации, анализ предметной области (выявление потребности в создании или модернизации комплекса программ с определенными функциями и качеством);
2. Разработка исходных требований для технического задания на проект ПС (упорядочивание и ранжирование приоритетов требований, координация функциональных требований с конструктивными и с возможностью их эффективной реализации);
3. Предварительный анализ и моделирование процессов обработки данных (от установления базовых отношений между понятиями, через определение интерфейсов доступа и атрибутов, к проекту модели состояний и взаимодействий между реальными объектами и процессами ПС);
4. Концепция создаваемого комплекса программ (первый исходный документ, согласуемый с заказчиком для создания комплекса программ. На основе этого описания формируется предварительное техническое задание на систему и её основные компоненты);
5. Стратегическое планирование проекта (долгосрочные цели развития ЖЦ ПС определенного функционального назначения, отражающие предварительные проекты всего будущего ЖЦ ПС, обеспечения их качества, защиты и безопасности функционирования, верификации и тестирования, управления конфигурацией и сопровождения);
6. Системный проект программного средства новой или модернизированной системы (полные требования к функциям и характеристикам качества комплекса программ, описание и графическое представление его архитектуры, БД и взаимодействия компонентов, предполагаемую модель ЖЦ, предварительные планы последующих этапов и работ; также проекты технического задания и контракта на детальное проектирование и весь ЖЦ ПС).
7. Организация документирования программных средств. Типы документаций.

Свойства документации:

1. Полнота;
2. Корректность;
3. Согласованность.



1. Организация документирования:
2. Вырабатывание стратегии и регламента;
3. Формирование профиля стандартов;
4. Адаптация стандартов в нормативные документы.

Профиль документов формируется на каждом этапе ЖЦ ПС. На начальном этапе проектирования следует выделять и формировать целесообразный комплект шаблонов документов, обеспечивающих регламентирование всех этапов, процессов и документов.

Типы документации:

1. Технологическая – для специалистов;
2. Эксплуатационная – для заказчика и пользователей, сопровождения;
3. Исследовательская – для руководителей, разработчиков и исследователей.
4. 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 ПС и БД регламентируются рядом процедур и документов, стандартизированных в ISO 14764 (сопровождение ПС), который детализирует требования к процессам переноса, определенным в базовом стандарте на ЖЦ ПС (ISO 12207). Специалисты, которые проводят перенос по рекомендациям этих стандартов, должны разработать план переноса, известить пользователей, обучить персонал, выдать предупреждения о завершении переноса, оценить влияние новой версии и внешней среды и архивировать соответствующие данные. Если систему или программный продукт (включая данные) переносят из старой в новую эксплуатационную среду, следует обеспечить, чтобы программный продукт и данные были корректно изменены при переносе. Для этого необходимо решить основные задачи:

● определить все добавляемые или изменяемые программные компоненты, продукты или данные;

● проверить соответствие реализации конкретных задач, спецификациям требований заказчика на перенесенную версию ПС и БД.

Для контроля переноса системы необходимо разработать, документально оформить и выполнить план переноса программного продукта. В содержание плана должны быть включены:

● анализ и формирование требований к результатам переноса;

● разработка (или приобретение) инструментальных средств для выполнения переноса;

● настройка программного продукта и данных к новым условиям и среде эксплуатации;

● выполнение процессов переноса;

● верификация и тестирование результатов переноса;

● обеспечение последующей поддержки прежней среды и программного продукта.

Разработка плана переноса должна быть основана на исходных данных и требованиях заказчика или потенциальных пользователей. После завершения сопроводителем планирования переноса, заказчику и пользователям должно быть направлено уведомление о планах и работах по переносу программного продукта и базы данных. В содержание уведомления целесообразно включить: объяснение того, почему прежнюю среду ПС нельзя больше сопровождать и поддерживать; описание новой среды и программного продукта с указанием даты, с которой они доступны для заказчика и пользователей.

Сопроводитель должен также предоставить пользователям план, процедуры и график переноса и решить

следующие задачи:

● определить все компоненты, затрагиваемые переносом;

● отработать обратную связь и информацию с заказчиком и пользователями;

● определить специфику пользователей;

● опубликовать график переноса.

Для плавного перехода в новую среду, пользователями параллельно могут выполняться работы в прежней и новой среде с соответствующими версиями ПС и БД. Сопроводитель должен выполнить следующие работы по обучению персонала:

● определить требования по обучению для реализации переноса;

● запланировать реализацию требований по обучению переноса;

* выполнить проверку результатов обучения после выполнения переноса;

● обновить и откорректировать планы обучения.

После завершения запланированного переноса должны быть посланы соответствующие уведомления всем заинтересованным сторонам. Все связанные с прежней средой документы, журналы регистрации и программы следует поместить в архивы. После завершения переноса целесообразно провести итоговый анализ для оценки влияния перехода к новой аппаратно-операционной среде на различные аспекты эксплуатации перенесенного программного продукта. Результаты анализа должны быть разосланы соответствующим заинтересованным сторонам для информации, руководства и использования в работе.

Отчетными результатами работ по переносу ПС и БД являются:

● перенесенный программный продукт на новой платформе;

● план реализации переноса;

● инструментальные средства для переноса;

● извещения о намерениях по переносу;

● уведомление о завершении переноса;

● архивные данные процессов и результатов переноса.

Затраты и сложность переноса информации базы данных зависят прежде всего от ее характеристик, которые отражают форматную, лингвистическую и физическую совместимость содержания переносимой БД между рассматриваемыми платформами:

● форматная совместимость характеризуется степенью соответствия данных в БД анализируемых платформ требованиям стандартов на форматы представления данных для документальных, фактографических, словарных или иных БД;

● лингвистическая совместимость определяется степенью использования в рассматриваемых БД единых лингвистических средств (классификаторов, рубрикаторов, словарей), формализованных соответствующими стандартами;

● физическая совместимость заключается в степени соответствия кодировки информации БД одинаковым стандартам на машиночитаемые носители информации.

Для средних и крупных проектов системный анализ переноса целесообразно завершать оценкой суммарной трудоемкости и длительности переноса программного продукта и их сопоставлением с полной разработкой ПС и БД при некотором использовании алгоритмического и системного задела. Кроме того, полученный комплекс программ следует оценивать по использованию памяти и производительности новой ЭВМ. Такую оценку необходимо проводить с учетом возможного тиража ПС и перспективы длительной его эксплуатации.

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

**Верификация** — процесс определения, выполняют ли программные средства и их компоненты требования, наложенные на них в последовательных этапах ЖЦ ПС. Важнейшей частью являются: анализ, обзоры и тестирование от требований. Основная цель верификации ПС - обнаружение, регистрация и устранение дефектов и ошибок, внесённых во время разработки и модификации программ. Верификация должна быть интегрирована как можно раньше с процессами проектирования, разработки и сопровождения. Обычно проводится сверху-вниз (от общих требований до детальных).

**Назначение верификации ПС** — последовательно проверить, что:

● общие требования к ИС, корректно переработаны в спецификацию требований высокого уровня к комплексу программ, удовлетворяющих исходным системным требованиям;

● в спецификации требований к функциональным компонентам низкого уровня, удовлетворяющих требованиям высокого уровня, требования высокого уровня правильно переработаны в архитектуру ПС;

● архитектура ПС и спецификации требований к компонентам низкого уровня корректно переработаны в исходные тексты программных и информационных модулей;

● исполняемый объектный код удовлетворяет требованиям к исходному тексту программных компонентов.

Верификации на соответствие спецификации требований на конкретный проект ПС также подлежат требования к технологическому обеспечению ЖЦ ПС и требования к эксплуатационной и технологической документации.

**Цели верификации ПС** достигаются посредством последовательного выполнения комбинации из просмотров, анализов, разработки тестовых сценариев и процедур и последующего выполнения этих процедур.

**Тестирование**:

В качестве объекта тестирования выступают функциональные требования, требования к безопасности, удобству применения, взаимодействия компонент между собой и внешней средой и несколько особняком стоит тестирование, связанное с изменениями:

Далее рассмотрим классификацию тестирования по уровням:

● Компонентное (модульное) тестирование (Сomponent/Unit testing)

● Интеграционное тестирование (Iintegration testing)

● Системное тестирование (System/end-to-end testing)

● Приемочное тестирование (Acceptance Testing)

Модульное или компонентное тестирование направлено на поиск дефектов и ошибок в небольших частях ПО: модулях, функциях, классах и т.д. Интеграционное тестирование предназначено для проверки взаимодействия между компонентами, ошибок в межмодульном интерфейсе, а также взаимодействия с операционной системой, аппаратурой, внешней средой. Основной задачей системного тестирования является проверка всех функциональных, требований в системе в целом; оно может выявить непроверенные сочетания входных данных, применение ПО в системе для этого непредусмотренной, несогласованное или неадекватное использование ресурсов, отсутствующую или неверная функциональность, неудобство использования и т.д. Приемочное тестирование определяет, удовлетворяет ли система приемочным критериям.

Достаточно часто используют **терминологию, отражающую знание тестировщика о системе**:

● Тестирование чёрного ящика (black box): тестировщик имеет доступ к ПО только через интерфейс; обычно критерий покрытия складывается из покрытия структуры входных данных и покрытия требований.

● Тестирование белого ящика (white box): разработчик теста имеет доступ к исходному коду; чаще всего применяется в случае модульного тестирования; используются метрики покрытия кода

● Тестирование серого ящика (grey box): разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется.

Также тестирование может быть классифицировано **по степени автоматизации:**

● Ручное тестирование (manual testing): ручное выполнение тестов и проверка результатов;

● Автоматизированное тестирование (automated testing): использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс;

● Полуавтоматизированное тестирование (semiautomated testing): смесь.

Одной из главных проблем автоматизированного тестирования является его трудоемкость: большие ресурсы могут тратиться на обновление самих тестов. Как правило автоматизированное тестирование является лишь дополнением ручного тестирования и не может его заменить полностью.

Существует классификация по степени формализации процесса тестирования:

● Тестирование по документации (formal testing)

● Тестирование ad hoc или тестирование без определенного плана и документации, основанное на импровизированном подходе к поиску ошибок (adhoc testing)

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

Структура документации и формы отдельных документов, используемых для конфигурационного управления и сопровождения программ, должны позволять точно документально описывать и идентифицировать каждую оформленную версию программных компонентов и ПС в целом в любое время на всем протяжении их ЖЦ. Стандарт ISO 9126 (ИТ. Оценка программной продукции. Характеристики качества и руководства по их применению) регламентирует характеристики качества программного продукта, его изменений и документов.

Методика оформления отчетов о выявленных дефектах, ошибках и предложениях по корректировке версий ПС должна содержать рекомендации испытателям и пользователям по выявлению, регистрации и формализации условий проявления и содержания дефектов и желательных модификациях испытываемых и/или эксплуатируемых версий. Эта методика должна включаться в состав эксплуатационной документации и передаваться каждому пользователю версии программного продукта. В методике следует стимулировать специалистов анализировать, подготавливать и представлять рекомендации заказчику и разработчикам по совершенствованию и развитию функций и качества поставляемой версии ПС. Результаты анализа и предложения необходимо передавать в унифицированной форме Отчетов специалистов о выявленных дефектах и предложениях по корректировке ПС, которые должны содержать:

● подробное описание сценария тестирования и исходных данных, при которых выявлен дефект;

● описание проявления дефекта и документы результатов его регистрации;

● предположение о возможной причине, вызвавшей проявление дефекта;

● предложение о возможной модификации ПС и его компонентов для устранения дефекта или совершенствования качества функционирования комплекса программ.

Изменения, отобранные аналитиками сопровождения и управления конфигурацией, переводятся из описаний предварительных изменений в другую часть базы данных. Здесь изменения конкретизируются вплоть до текстов корректировок программ или созданных новых компонентов. Кроме того, для каждой подготовленной корректировки следует регистрировать результаты ее рассмотрения, проверки и утверждения на введение в новую базовую версию ПС или для частных извещений пользователям.

Отвергнутые корректи ровки возвращаются в состав предлагаемых изменений.

Описание выявленных дефектов и предложений по совершенствованию функций комплекса программ, а также результатов анализа предполагаемых корректировок версий ПС должен содержать отчеты пользователей о выявленных дефектах и предложениях и дополнительно:

● оценки сложности, трудоемкости, эффективности и срочности модификаций программ;

● оценки возможного влияния предлагаемых изменений на эксплуатацию версий ПС, имеющихся у пользователей.

Все корректировки, утвержденные для введения в очередную версию, следует регистрировать в отдельной подсистеме базы данных. Для каждого изменения должны документироваться содержательная аннотация, а также общие характеристики и достигнутые на предварительных испытаниях показатели качества очередной версии программного продукта. Результаты испытаний версии запоминаются вместе с условиями и параметрами, при которых они формировались, а также с набором тестов или указаниями места их хранения.

Описание подготовленных и утвержденных корректировок, а также реализованных изменений и обобщенных характеристик модифицированной базовой версии программного продукта должно содержать:

● причину изменения программ и базы данных (ошибка, дефект, совершенствование);

● содержание изменений программ и базы данных, а также документации на версию ПС или компонента;

● результаты квалификационного тестирования базовой версии программного продукта с предполагаемыми изменениями;

● результаты испытаний и обобщенные характеристики качества базовой версии программного продукта после внесения изменений;

● решение по распространению пользователям проведенной модификации или версии программного продукта;

● адрес хранения корректировок, документов и квалификационных тестов новой базовой версии программного продукта.

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

ТЭ-обоснование проекта – обоснование его рентабельности на основе рассчитанных технико-экономических показателей.

Формирования требований к функциональной пригодности ПС наиболее ответственная, стратегическая задача начальных этапов технико-экономического обоснования проекта.

Основные ТЭП:

● Трудозатраты;

● Время;

● Число специалистов.

ТЭ для ЖЦ разделяет его на две части:

● ТЭ анализа и конструирования – зависит от функциональных характеристик объекта и технологий;

● ТЭ эксплуатации, сопровождения и т.д. – зависит от массовости, потребительских характеристик, внешних факторов.

Данные для прогнозирования ТЭП:

● Функции самого объекта, средства автоматизации труда;

● Экономические характеристики прототипов, пилотных проектов в той же области, обобщенный опыт. Используется экспериментальное распределение основных ТЭП по этапам работ и времени реализации компонентов, а также база исходных данных реальных завершенный разработок.

Необходимо привлекать заказчика к оценке ТЭП и производить учет действительных и оценочных затрат для:

● Корректировка оценок и учет ошибок;

● Корректировка методов оценивания;

● Обновление оценок затрат при изменении характеристик ПС;

Измерение размера ПС является ключевым фактором в оценке ТЭП, далее по вычисленным ТЭП формируется график работ.

Исходные данные для оценки ТЭП новых проектов в общем случае:

● обобщенные характеристики использованных ресурсов и технико-экономические показатели завершенных разработок;

● реализованные и обобщенные перечни выполненных работ, и реальные графики проведенных ранее разработок;

● цели и содержание частных работ в процессе создания сложных комплексов программ и требования к их выполнению;

● структура и содержание документов частных работ.

Группы факторов, влияющие на ТЭП при разработке программ:

● требования к его функциональным характеристикам и к качеству;

● организация процесса разработки;

● технологическая среда;

● аппаратурные вычислительные средства.

Методика 1

● экспертная оценка минимального числа факторов;

● сформулированы концепция и ТЗ с достоверностью 20-40%.

Цель – решение о продолжении разработки ПС, либо изменение, некоторых ТЭП и выделяемых ресурсов, либо прекращение проектирования данного ПС.

При наличии перечисленных исходных данных и положительной оценке целесообразности экспертного анализа ТЭП проекта может реализовываться методика:

● определение класса ПС, сложности, функций;

● экспертная оценка размера – числа строк с учетом повторно-используемого кода и языка;

● оценка стоимости и длительности одно LOC (строки);

● расчет полных трудозатрат и длительности, среднего числа специалистов;

● обобщение основных ТЭП и полной стоимости, ТЭ-обоснование рентабельности.

Базовый уровень оценки масштаба:

● приемлемый минимум функций и требований для заказчика;

● разумная вероятность успеха с точки зрения возможностей и ресурсов разработчиков.

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

Цель планирования ЖЦ программного средства состоит в выборе и определении способов создания и совершенствования ПС, которые способны удовлетворить требованиям технического задания, спецификаций и контракта, а также обеспечить уровень качества, соответствующий заданным требованиям. В современных стандартах подчеркивается, что эффективное планирование - определяющий фактор высокого качества всего ЖЦ программного средства, удовлетворяющего требованиям заказчика. В стандартах ISO 12207 (ИТ. Системная и программная инженерия. Процессы ЖЦ ПС) и ISO 16326 (Разработка ПО. Руководство по применению ISO/IEC 12207 к управлению проектом) рекомендуется определить администратора, который должен подготовить планы для выполнения процессов ЖЦ ПС. Планы, связанные с выполнением процесса, должны содержать описания соответствующих работ и задач, обозначения создаваемых компонентов программного средства и охватывать следующие задачи:

● установление графиков своевременного решения частных задач и всего ПС;

● оценки необходимых трудозатрат на задачи и проект в целом;

● определение ресурсов, необходимых для выполнения задач и проекта;

● распределение задач по исполнителям;

● определение обязанностей исполнителей;

● определение критических ситуаций, связанных с задачами или процессами ЖЦ ПС;

● установление используемых в процессах ЖЦ ПС критериев управления качеством;

● определение затрат, связанных с реализацией каждого процесса;

● обеспечение условий и определение инфраструктуры выполнения процессов ЖЦ ПС.

Оценки проекта, используемые при планировании, должны охватывать:

● стоимость реализации соответствующих процессов;

● инфраструктуру обеспечения реализации процессов;

● потребности в ресурсах, включая соответствующее управление и контроль;

● оценку и контроль качества реализации процессов;

● управление риском результатов процессов;

● обеспечение среды программной инженерии проекта ПС;

● задания, выполняемые в каждом процессе и (или) работе.

Администраторы каждого программного проекта должны стремиться по возможности использовать существующую организационную инфраструктуру предприятия. Если существующая инфраструктура не удовлетворяет потребностям конкретного проекта, тогда она должна быть соответствующим образом адаптирована или дополнена. Для устранения несовершенства (неполноты) существующей инфраструктуры может потребоваться использование субподрядных работ.

Планирование является постоянной, повторяющейся работой, при выполнении которой оценивают, уточняют и, при необходимости, корректируют ход проекта. Администраторы программного проекта должны использовать соответствующие процессы, способствующие проведению перепланирования и уточненных оценок в ЖЦ программного проекта. В каждом программном проекте имеется множество взаимосвязей, поэтому для уточнения исходного плана управления проектом обычно требуется несколько итераций. При необходимости внесения в план управления информации, содержащейся в других планах, в данном плане может быть приведена ссылка на эти планы.

Стандартами ISO 16326 и ISO 90003 (Руководящие указания по применению ИСО 9001:2008. Системы менеджмента качества. Требования при разработке программных продуктов).

Рекомендуется в процессе планирования ЖЦ ПС подготовить и утвердить содержание следующих планов:

● разработки компонентов и всего ЖЦ ПС, который должен определять используемую модель ЖЦ комплекса программ и его компонентов, а также внешнюю технологическую среду проектирования;

● верификации и тестирования, который определяет методы и средства, способные удовлетворить последовательные цели процесса устранения дефектов и контроля качества ПС и его компонентов;

● реализации процессов интеграции компонентов в версии комплекса программ;

● сопровождения и управления конфигурацией ПС, который должен устанавливать методы и средства, при помощи которых будут удовлетворяться цели процесса совершенствования, управления изменениями и корректировками программ;

● тиражирования, адаптации и внедрения версий ПС для конкретных пользователей, включая их подготовку и обучение;

● документирования процессов и результатов ЖЦ ПС, создания и выпуска технологической и эксплуатационной документации;

● управления и обеспечения качества ПС, определяющих методы и средства, при помощи которых будет гарантировано требуемое качество комплекса программ.

Соответствующие планы должны быть разработаны администраторами вспомогательных процессов, так как эти процессы обычно являются частью проекта. Данные планы должны быть привязаны к базовому плану управления ЖЦ программного проекта и обеспечивать его реализацию; они могут быть оформлены в виде отдельных планов или включены в общий план. Планы должны быть согласованы (утверждены) менеджером проекта программного средства и подлежат контролю при внесении изменений в проект.

При проектировании ПС **планирование процессов управления качеством ПС** целесообразно отделять от непосредственного планирования и управления процессами создания и совершенствования крупных комплексов программ. Мероприятия планирования, обеспечивающие качество программ, должны охватывать не только завершающие испытания, но и весь жизненный цикл ПС.

Организационной основой управления качеством ПС на базе стандартов ЖЦ является **план обеспечения заданных характеристик качества** на всех этапах жизненного цикла комплекса программ. Для этого до начала разработки в процессе формирования требований технического задания следует сформулировать основные положения методики обеспечения качества, поэтапных испытаний компонентов и определения достигнутых значений характеристик, допустимых для продолжения работ на следующих этапах.

В плане управления качеством ПС должны быть отражены:

● цели управления качеством, номенклатура и требования к значениям характеристик качества, область действия требований и условия их применения;

● методы управления и достижения заданных значений качества; процедуры, которые должны выполняться для каждого процесса и на протяжении всего ЖЦ ПС; действия, связанные с отчетностью об ошибках, с трассировкой и системой корректирующих действий;

● организация разработчиков и технология создания ПС; утвержден ные обязанности специалистов по обеспечению качества, их ответственность и полномочия на утверждение программных компонентов и документов;

● ресурсы, базовые документы и стандарты, используемые для обеспечения качества на всех этапах разработки;

● средства автоматизации разработки, обеспечивающие достижение и измерение заданных свойств и значений характеристик качества;

● структура и содержание отчетных документов, удостоверяющих достижение определенного качества компонентов и ПС в целом на последовательных этапах разработки, а также их соответствие стандартам и требованиям заказчика.

Управление качеством комплексов программ предполагает формализацию технологии обеспечения их жизненного цикла, а также выделение в специальный процесс поэтапного измерения и анализа текущего качества программных компонентов и проекта в целом.

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

Формализация требований – процесс, в ходе которого вырабатывается и обеспечивается соглашение между заказчиком и выполняющими проект специалистами.

Процессы:

Выявление интересов заказчика:

● достигнуть соглашения между заказчиком и разработчиком по определению проблемы, целей и задач проекта;

● выделить основные причины - проблемы, являющиеся их источниками и стоящие за основной проблемой проекта системы и ПС;

● выявить заинтересованных лиц и пользователей, чье коллективное мнение и оценка в конечном итоге определяет успех или неудачу проекта;

● определить, где приблизительно находятся область и границы возможных решений проблем;

● понять ограничения, которые будут наложены на проект, команду и решения проблем.

Выявление интересов пользователей и др. заинтересованных лиц:

● интервьюирование и анкетирование, проведение интервью, формулирование 10 - 15 наиболее часто упоминавшихся потребностей заказчика и пользователей;

● совещания, посвященные анализу и синтезу требований - формулирование и определение целей программного продукта; обязательно убедиться в согласии заказчиков;

● мозговой штурм и отбор идей, чтобы: выявить и/или уточнить функции проекта, определить приоритеты, риски, трудоемкости реализации функций;

● анализ иллюстративных прецедентов в приложении к концепции требований (или системному проекту), чтобы их функции были наглядны и понятны;

● по возможности выявление или создание временных прототипов на основе первичных требований.

На этапе проектирования последовательно должны определяться требования:

● концепции – к назначению, функциональной пригодности и номенклатуре конструктивных характеристик;

● предварительное проектирование – к шкалам и мерам применяемых атрибутов характеристик качества с учетом общих ограничений ресурсов;

● детальное проектирование – подробные требования к атрибутам качества с детальным учетом и распределением реальных ограниченных ресурсов.

Факторы требований к характеристикам качества:

● функциональная пригодность ПС;

● возможные конструктивные характеристики качества комплекса программ для улучшения функциональной пригодности;

● доступные ресурсы.

Разработка концепции: выявление класса, назначения и функций ПС.

ISO 12182, характеристики качества ISO 9126

Разделить потребности по приоритетам требования качества:

● заказчик – требования в контракте и ТЗ;

● пользователи – функциональная пригодность, надежность, корректность и т.д.;

● разработчики – ясность и корректность описаний требований к функциям, архитектуре, интерфейсам;

● специалисты по сопровождению и модификации – характеристики, поддерживающие сопровождение;

● специалисты развертывания и реализации – мобильность.

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

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 ЖЦ ПС не всегда удается достигнуть требуемого положительного эффекта и может проявляться некоторый ущерб – риск в создаваемых проектах, программных продуктах и их характеристиках. Риски проявляются, как **негативные последствия дефектов функционирования** и применения ПС, которые способны вызвать ущерб системе, внешней среде или пользователю, в результате отклонения характеристик объектов или процессов, от заданных требованиями заказчика, согласованными с разработчиками.

Возможные риски в ЖЦ сложных ПС:

● Срыв сроков проекта.

○ Провести анализ сроков;

○ Оценка стоимости (пересматривается анализ стоимости);

○ Повторное использование готовых компонентов;

● Ошибки проектирования.

○ Провести анализ разработчиков;

○ Провести анализ ещё раз;

○ Формулировка дополнительных условий ;

○ Ранние пользовательские прототипы;

● Разработка ошибочного интерфейса (плохая связь с заказчиком).

○ Обсудить с заказчиком стилевые и функциональные особенности интерфейса.

● Потеря прибыльности (неумение заключать договор, не учли конкурентов на рынке).

○ Снижать требования к продукту (сократить функции продукта => снизится стоимость продукта).

**Ошибка в программе** - неправильность, погрешность или неумышленное искажение объекта или процесса, что может быть причиной ущерба — риска при разработке. При этом предполагается, что известно правильное, эталонное состояние объекта или процесса, по отношению к которому может быть определено наличие отклонения — ошибки или дефекта. Если нет эталона для проверки модели то проверяют на специфакациях, прописанных в ТЗ.

**Типы ошибок и дефектов**:

● Первичные в программах и данных — потенциальные угрозы проекту программного средства

● Вторичные – Фиксируются при тестировании.

Вторичные ошибки являются определяющими для эффективности функционирования программ, однако не каждая первичная ошибка вносит заметный вклад в выходные результаты.

Как оценивается степень ошибок:

● «1-3 приоритет» - небольшие ошибки. Они не влияют отрицательно на работу программы.

● «4-7 приоритет» - умеренные, средние ошибки. Влияют конечного пользователя, но имеют малые последствия.

● «8-10 приоритет» - критические ошибки. Ошибки с высоким влиянием. Напр. сбой всей системы. Потеря важных данных. Крах системы, не помогает “restart”.

Первичные ошибки в ПС в порядке уменьшения их влияния на сложность обнаружения и масштабы корректировок можно разделить на следующие группы:

● ошибки, обусловленные сложностью компонентов и ПС в целом;

● ошибки вследствие большого масштаба комплекса программ и завышенных требований качества;

● ошибки планирования и корректности требований модификаций;

● ошибки проектирования;

● системные ошибки, обусловленные отклонением функционирования ПС в реальной системе;

● алгоритмические ошибки, связанные с неполным формированием необходимых условий решения и некорректной постановкой целей функциональных задач;

● ошибки реализации спецификаций изменений – программные дефекты, возможно, ошибки нарушения требований или структуры компонентов ПС;

● программные ошибки, в исходных кодах или при трансляции в объектный код;

● ошибки в документации;

● технологические ошибки подготовки физических носителей и документации, установки ПС на оборудование, вывода результатов работы.

На всякий случай (IEEE 729 Словарь программной инженерии. Термины): Ошибка - действия человека, приводящие к дефектам в софте. Дефект - проявление ошибки в софте. Сбой - прекращение способности функционального модуля выполнять требуемую функцию

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

**Объектно-ориентированное проектирование (ООП)** предназначено организовывать программные системы с большими базами данных на основе описаний объектов реального мира, важных для пользователей. Объекты реального мира, существующие внутри области действия ООП программных проектов, определяются в терминах целей, характеристик и ответственностей поведения соответствующих данных (атрибутов) и отношений с другими многочисленными объектами. Все функции при этом скрыты внутри деталей описаний объекта.

Под процессом ООП подразумевается проектирование классов объектов и взаимоотношений между этими классами. Объектно-ориентированные системы можно рассматривать как совокупность автономных объектов.

Общий процесс объектно-ориентированного проектирования состоит из нескольких этапов:

● определение рабочего окружения системы и разработка моделей ее использования;

● проектирование архитектуры программной системы;

● определение и идентификация основных объектов системы;

● разработка модели архитектуры комплекса программ;

● определение и документирование интерфейсов объектов.

**Главное преимущество ООП** программных средств состоит в том, что оно упрощает задачу внесения изменений в системную архитектуру, поскольку представление состояния объекта не оказывает на нее влияния.

Основные особенности ООП:

● основные компоненты программной системы представляются как объекты со своими состояниями и операциями;

● объекты предоставляют сервисы (методы) другим объектам и создаются в реальном времени на основе определения класса объектов;

● объекты могут быть реализованы последовательно и параллельно, параллельный объект может быть пассивным, у которого состояние изме няется только через его интерфейс, или активным, который может изменять свое состояние без вмешательства извне;

● в процессе объектно-ориентированного проектирования возможно создание ряда различных моделей, которые можно разделить на статические (модели классов, модели обобщения, модели агрегирования) и дина мические (модели последовательностей, модели конечного автомата);

Использование методов ООП строго регламентировано, поэтому:

● возрастает производительность труда разработчиков благодаря переходу к высокоэффективному методу — на базе предварительного анализа проекта;

● запросы и объекты реального мира проще моделируются путем концентрации внимания на классах, а не на алгоритмах их функционирования;

● компоненты системы легко изменяются и применяются повторно;

● требования проще отслеживаются;

● поддерживается эффективное прототипирование;

● разработка проекта отличается непрерывностью в представлении объектов — одни и те же типы диаграмм применяются как при анализе, так и на этапе разработки;

● работа по проектированию может осуществляться с помощью универсальных технологических инструментов.

ООП — часть объектно-ориентированного процесса разработки системы. Этот подход подразумевает выполнение трех этапов:

● объектно-ориентированный анализ

● объектно-ориентированное проектирование;

● объектно-ориентированное программирование.

Использование объектного подхода повышает уровень унификации разработки и пригодность для повторного использования не только программных компонентов, но и больших комплексов программ, что ведет к созданию унифицированной среды разработки и переходу к сборочному созданию программных продуктов. Взаимосвязью между структурным и объектно-ориентированным подходами является общность ряда категорий и понятий жизненного цикла ПС.

В программных средствах при ООП рекомендуется выделять три уровня:

● уровень интерфейсов, который занимается всеми взаимодействиями с другими частями системы и предоставлением внешних интерфейсов системы;

● уровень сбора данных, управляющий сбором информации из внешней среды и обобщающий данные перед отправкой их в систему построения обобщенных результатов;

● уровень объектов, в котором представлены и описаны все объекты, используемые в процессе сбора исходных данных.

**Основные парадигмы программирования:**

● процедурное: последовательно выполняемые операторы можно собрать в подпрограммы (C, Pas);

● объектно-ориентированное: см. выше (C++, C#, Smalltalk);

● функциональное: процесс вычисления трактуется как вычисление значений функций в математическом понимании последних (Lisp, Scheme, Haskell);

● логическое: основано на теории и аппарате математической логики с использованием математических принципов резолюций (Пролог).

Процедурное и ООП являются разновидностями **императивного** программирования (исходный код - последовательные инструкции, оперирующие данными в памяти == как решать), ФиЛП - **декларативного** (исходный код - спецификация, описание решения и результата == что получить).

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

**Доступные ресурсы обеспечения жизненного цикла ПС** включает реальные финансовые, временные, кадровые и аппаратурные ограничения затрат, в условиях которых происходит создание и совершенствование комплексов программ. В результате доступные ресурсы становятся косвенными критериями или факторами, влияющими на выбор методов разработки, на достигаемые качество и эффективность применения программных продуктов. Поэтому одной из основных задач при проектировании ПС является экономический анализ и определение необходимых ресурсов для создания и обеспечения всего ЖЦ ПС в соответствии с требованиями контракта и технического задания.

Наиболее общим видом ресурсов, используемых в жизненном цикле ПС, являются допустимые **финансово-экономические затраты** или эквивалентные им **величины трудоемкости** соответствующих работ. При разработке, тестировании и анализе качества этот показатель может применяться или как вид ресурсных ограничений, или как оптимизируемый критерий, определяющий целесообразную функциональную пригодность ПС. При этом необходимо также учитывать затраты на разработку, закупку и эксплуатацию системы качества, на технологию и комплекс автоматизации проектирования программ и баз данных, которые могут составлять существенную часть совокупной стоимости и трудоемкости разработки и всего ЖЦ ПС.

Затраты в жизненном цикле ПС определяются не только этапами разработки, но и этапами эксплуатации и сопровождения.

**Время** разработки определенных компонентов и версий ПС является невосполнимым ограниченным ресурсом реальных проектов. Этот ресурс определяет достижимое качество сложных комплексов программ в процессе их разработки и сопровождения.

Доступные разработчикам ПС **вычислительные ресурсы** объектных и технологических ЭВМ являются одним из важнейших факторов, определяющим достижимое качество сложных ПС. В процессе проектирования целесообразно выделять определенные ресурсы ЭВМ на оператив ное обеспечение качества, повышение защищенности, безопасности и надежности функционирования. Допустимая величина и рациональное распределение ресурсов ЭВМ на отдельные методы улучшения определенных конструктивных хар-к качества ПС оказывают существенное влияние на достигаемые ими значения.

Обобщенными ресурсами ЖЦ проекта ПС являются доступные стоимость или совокупные трудовые, временные и материальные затраты, необходимые для приобретения, создания, модификации и эксплуата ции компонентов и всего комплекса программ. Эти характеристики непосредственно влияют практически на все показатели качества и определяют рентабельность покупки или создания заново конкретного программного продукта. Качество является относительным понятием, которое зависит от ресурсов и субъектов, осуществляющих его оценку с позиции эффективности использования, а также от состояния рынка соответствующей продукции, ее производителей и технологий.

Важнейшим ресурсом при создании программных средств являются люди — специалисты, с их уровнем профессиональной квалификации, а также с многообразием знаний, опыта, стимулов и потребностей. Быстрый рост сложности и повышение ответственности за качество комплексов программ привели к появлению новых требований к специалистам программной инженерии, обеспечивающим все этапы жизненного цикла ПС.

Для рационального распределения этих ресурсов необходимо знать, как отращивается изменение затрат на улучшении каждой характеристики качества ПС.

**Обеспечение функциональной пригодности** является основной целью при использовании финансовых, трудовых, вычислительных и других ресурсов в ЖЦ ПС. Необходимость выполнения ряда требований к остальным, конструктивным хар-кам качества часто приводит к тому, что исполь зование ресурсов на их реализацию может превышать базовые затраты на обеспечение функциональной пригодности. В то же время затраты на выполнение этих требований всегда направлены на повышение и совершенствование функциональной пригодности. Поэтому обычно трудно четко выделить и количественно оценить все виды затрат, используемых только на функциональную пригодность.

Для анализа затрат ресурсов в ЖЦ ПС при проектировании их целесообразно разделить на 2 части:

● затраты на создание программных компонентов, обеспечивающих базовые свойства функциональной пригодности комплекса программ для его применения по прямому назначению пользователями, в соответствии с требованиями контракта и технического задания;

● основные составляющие дополнительных затрат, обеспечивающие требуемые конструктивные характеристики качества для улучшения функциональной пригодности ПС в соответствии с целями и сферой его применения.

Кроме того, следует учитывать совокупность затрат ресурсов на технологию, инструментарий автоматизации разработки и систему качества, обеспечивающие жизненный цикл ПС. Эта составляющая затрат зависит не только от хар-к проектируемого ПС, но и от интенсивности применения технологических средств.

Затраты на обеспечение функциональной пригодности зависят, в первом приближении, от сложности алгоритмов, объема комплекса программ и баз данных, которые определяют затраты труда и длительность полного цикла их разработки. Основные затраты идут на совеществленный, преимущественно интеллектуальный, труд специалистов различных категорий. Наиболее универсальной единицей стали трудозатраты специалистов в **человеко-днях** или **человеко-месяцах**, которые обычно достаточно просто могут преобразовываться в стоимость процесса разработки.

Обычно совершенствование качества и повышение затрат на реализацию характеристик способствует снижению затрат при их эксплуатации. Последние трудно оценить априори, так как они зависят от внешней среды и активности применения конкретного ПС, а не от его свойств и требуемого качества.

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

Управление конфигурацией включает действия и средства, позволяющие устанавливать категории,

статус и личности руководителей, которые правомочны определять целесообразность и эффективность изменений, а также техническую реализуемость корректируемых версий с учетом ограничений бюджетов и сроков

При анализе и селекции изменений важен точный учет степени влияния каждого изменения на все остальные компоненты и на их основные характеристики качества. Поэтому решения о кардинальных изменениях ПС и компонентов должны при ниматься на достаточно высоком уровне руководства проектом, способного оценить их влияние на концептуальную целостность и качество всей информационной системы

Цель управления конфигурацией при разработке и сопровождении сложных ПС и систем, состоящих из многих компонентов (единиц конфигурации), каждый из которых может иметь разновидности или версии, обеспечить управляемое и контролируемое развитие их структуры, состава компонен тов и функций, а также сокращение дефектов в течение всего ЖЦ ПС. В процессе организации конфигурационного управления необходимо построить и использовать компактные и наглядные схемы однозначной иерархической идентификации и изменения взаимодействия компонентов ПС:

● объектов - модулей и компонентов ПС, разного уровня интеграции, подвергающихся корректировкам (систему идентификации и адресации изменений в комплексе программ и в документах);

● корректировок содержания и взаимодействия проводимых изменений, которая должна обеспечивать возможность однозначного контроля, истории развития модификаций компонентов любого уровня, во времени и в пространстве элементов версий комплекса программ (типы, содержание и взаимосвязь корректировок);

● специалистов, участвующих в конфигурационном управлении и сокращении дефектов, их права на доступ к определенным компонентам ПС и документам на конкретных стадиях сопровождения, реализации и утверждения изменений.

Процесс управления конфигурацией, (стандарт ISO 12207 (ИТ. Системная и программная инженерия. Процессы ЖЦ ПС) п.6.2), является процессом применения административных и технических процедур на всем протяжении ЖЦ ПС для: обозначения, определения и установления состояния базовой версии программных продуктов в системе; управления изменениями и выпуском объектов; описания и сообщения о состояниях объектов и заявок на внесение изменений в них; обеспечения полноты, совместимости и правильности объектов; управления хранением, обращением и поставкой объектов.

Этот процесс включает: подготовку процесса; определение конфигурации; контроль конфигурации; учет состояний конфигурации; оценку конфигурации; управление выпуском и поставку программного продукта. Все основные и вспомогательные процессы подлежат адаптации и конкретизации применительно к характеристикам определенного проекта

Стандарт ISO 15846 (ИТ. Процессы ЖЦ ПО. Управление конфигурацией) обобщает, детализирует и развивает основные концептуальные положения, представленные в стандарте ISO 12207. Излагаются подробные рекомендации по реализации его базовых требований по управлению конфигурацией ПС. Существенным достоинством стандарта ISO 15846 является подробное и систематичное изложение практических рекомендаций по управлению конфигурацией сложных комплексов программ, которые целесообразно использовать в крупных современных реальных проектах систем.

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

**Первым** компонентом для системного анализа и требований к качеству является комплекс программ СУБД. Практически весь набор характеристик и атрибутов качества ПС, изложенный в стандарте ISO 9126 (ИТ. Оценка программной продукции. Характеристики качества и руководства по их применению), в той или иной степени, может использоваться при формировании требований к качеству СУБД. Во всех случаях важнейшими характеристиками качества СУБД являются требования к функциональной пригодности для процессов формирования и изменения информационного наполнения БД администраторами, а также доступа к данным и представления результатов пользователям БД. Ниже за основу принята номенклатура и содержание стандартизированных характеристик сложных комплексов программ, которые адаптируются применительно к понятиям и особенностям компонентов БД. В зависимости от конкретной проблемно-ориентированной области применения СУБД, приоритет при системном анализе требований к качеству может отдаваться различным, конструктивным характеристикам: либо надежности и защищенности применения (финансовая сфера), либо удобству использования малоквалифицированными поль зователями (социальная сфера), либо эффективности использования ресурсов (сфера материально-технического снабжения).

**Вторым** компонентом БД является собственно накапливаемая и обрабатываемая информация. В системах БД доминирующее значение приобретают сами данные, их хранение и обработка. Ниже сделан акцент на системный анализ требований и составляющих характеристик качества этого объекта - на информацию БД (ИБД) с предположением, что средства СУБД способны их обеспечить. Для оценивания качества информации БД может сохраняться общий, методический подход к выделению адекватной номенклатуры стандартизированных в ISO 9126 базовых характеристик и субхарактеристик качества ПС. Выделяемые показатели качества должны иметь практический интерес для пользователей БД и быть упорядочены в соответствии с приоритетами практического применения. Кроме того, каждый выделяемый показатель качества ИБД должен быть пригоден для достаточно достоверного оценивания или измерения, а также для сравнения с требуемым значением при испытаниях заказчиком. При проектировании каждой БД в контракте, техническом задании и в спецификации должны селектироваться и формализоваться представительный набор функциональных требований к качеству ИБД, адекватный ее назначению и области применения, а также требованиям заказчика и потенциальных пользователей. Так же как для ПС, характеристики качества ИБД можно разделить на функциональные и конструктивные. Их номенклатура, содержание и субхарактеристики ниже базируются на описаниях, рекомендуемых стандартом ISO 9126.

Функциональная пригодность ИБД может представлять сложную проблему для определения соответствия требований реальным значениям необходимых атрибутов качества, особенно, для больших распределенных БД при использовании разнообразной и сложной информации об анализируемых объектах. Мерой качества функциональной пригодности может быть степень покрытия целей, назначения и функций БД, доступной пользователям информацией. Так же как для ПС, для БД в составе функциональной пригодности целесообразно использовать группу субхарактеристик, определяющих функциональные и структурные требования к базам данных. Дополнительно функциональная пригодность многих ИБД может отражаться:·

● полнотой накопленных описаний объектов - относительным числом объектов или документов, имеющихся в БД, к общему числу объектов по данной тематике или по отношению к числу объектов в аналогичных БД того же назначения;·

● идентичностью данных - относительным числом описаний объектов, не содержащих дефекты и ошибки, к общему числу документов об объектах в ИБД;

● актуальностью данных - относительным числом устаревших данных об объектах в ИБД к общему числу накопленных и обрабатываемых данных.

К конструктивным характеристикам качества информации БД в целом можно отнести, с некоторым уточнением понятий, субхарактеристик и атрибутов, практически все стандартизированные показатели качества ПС, которые представлены в ISO 9126.

Требования к информации БД также должны содержать обеспечение ее надежности, эффективности использования ресурсов ЭВМ, практичности, применимости, сопровождаемости и мобильности. Содержание и атрибуты этих конструктивных характеристик в данном случае несколько отличаются от применяемых для программ, однако, их сущность, как базовых понятий и характеристик качества объектов, целесообразно использовать при проектировании для систематизации и регламентированного формирования требований к этим компонентам систем. Меры и шкалы для оценивания конструктивных характеристик, в значительной степени, могут применяться те же, что при анализе качества ПС.