ЛЕКЦИЯ 1

Тема: термины и определения стандартизации и сертификации. Цели и задачи стандартизации. Категории стандартов. Методика разработки и утверждения стандартов.

Цель: изучить основные понятия стандартизации

Мы поговорим о мониторинге и оценке ПО. Чем отличается и зачем нужно мониторить и

Современные контракты и планы на создание сложных программных средств (ПС) для информационных систем подготавливаются и оцениваются часто неквалифицированно, на основе неформализованных представлений заказчиков и разработчиков о требуемых функциях и их характеристиках качества.

Т.е. мы делаем, то что требует заказчик, а не следуем правилам, как нужно правильно проектировать и разрабатывать

Во многих случаях нужное качество программных средств зависит от интуиции, вкусов и квалификации их заказчиков и пользователей.

Значительные системные ошибки при определении требуемых характеристик качества, оценке трудоемкости, стоимости и длительности создания ПС являются достаточно массовыми и типичными.

Многие созданные программные продукты не способны полностью выполнять требуемые функциональные задачи с гарантированным качеством и их приходится долго и иногда безуспешно дорабатывать для достижения необходимого качества и надежности функционирования, затрачивая дополнительно большие средства и время.

В результате программные средства*, не соответствуют исходном* и *первоначальным требованиям заказчика к характеристикам качества, не укладываются в согласованные графики и бюджет разработки.*

Стандартизация — это *процесс установления и применения стандартов, под которыми понимается «образец, эталон, модель, принимаемые за исходные для сопоставления с ними других подобных объектов».*

Стандарт как нормативно-технический документ устанавливает комплекс норм, правил, требований к объекту стандартизации и утверждается компетентным органом.

Стандарт*-это уже документ по стандартизации, в котором описаны нормы и правила и требования к продукту. Содержит в себе показатель изменяемости качества.*

Применение стандартов способствует улучшению качества создаваемого изделия (в данном случае документа).

Документирование *— это процесс создания и оформления документа.* Государственный стандарт определяет документирование как «запись информации на различных носителях по установленным правилам»*.

СТАНДАРТИЗАЦИЯ. – это установление и применение правил с целью упорядочения деятельности в определённой области на пользу и при участии всех заинтересованных сторон, в частности, для достижения всеобщей оптимальной экономии при соблюдении условий эксплуатации и требований безопасности.

Стандартизация, основанная на объединённых достижениях науки, техники и передового опыта, определяет основу не только настоящего, но и будущего развития промышленности.

Из *определения* следует***, что стандартизация*** – это плановая деятельность по установлению обязательных правил, норм и требований, выполнение которых обеспечивает экономически оптимальное качество продукции, повышение производительности общественного труда и эффективности использования материальных ценностей при соблюдении требований безопасности.

СТАНДАРТ.– нормативно-технический документ по стандартизации, устанавливающий комплекс норм, правил, требований к объекту стандартизации и утверждённый компетентным органом.

Стандарты содержат показатели, которые гарантируют возможность повышения качества продукции и экономичности её производства, а также повышения уровня её взаимозаменяемости.

ЦЕЛИ И ЗАДАЧИ СТАНДАРТИЗАЦИИ.

Главная цель Государственной системы стандартизации (ГСС) – с помощью стандартов, устанавливающих показатели, нормы и требования, соответствующие передовому уровню отечественной и зарубежной науки, техники и производства, содействовать обеспечению пропорционального развития всех отраслей страны.

Цели Следить за тем, чтобы все отрасли развивались одинакова не нарушая нормы требования и показатели стандарта, соответствовали государственному и мировому уровню..

задачи *Госстандарта является:*

1) разработка мер по *повышению эффективности* стандартизации
2) *улучшение качества* выпускаемой продукции и экономичности её производства путём внедрения систем стандартов при комплексной
3) *развития межотраслевой унификации*

4) *создания общетехнических систем стандартов*, обеспечения единства и достоверности измерений в стране и др.

КАТЕГОРИИ СТАНДАРТОВ.

В зависимости от сферы действия ГСС (Государственной системы стандартизации**) предусматривает следующие категории стандартов**:

1) государственные (ГОСТ),

2) отраслевые (ОСТ),

3) республиканские (РСТ)

4) стандарты предприятий (СТП).

1. ***Государственные стандарты*** *обязательны для всех предприятий*, организаций и учреждений страны в пределах сферы их действия.

Государственные стандарты устанавливают требования к продукции массового и крупносерийного производства широкого и межотраслевого производства, к изделиям, прошедшим государственную аттестацию, экспортным товарам; *они устанавливают также общие нормы, термины* и т. п.

2. ***Отраслевые*** стандарты *используют все предприятия и организации данной отрасли* (например, IT –чистота и правильность написания кода), а также другие предприятия и организации, разрабатывающие, изготовляющие и применяющие изделия, которые относятся к номенклатуре, закреплённой за соответствующим министерством.

Отраслевые стандарты устанавливают требования к продукции, не относящейся к объектам государственной стандартизации, к технологической оснастке, инструменту, специфическим для отрасли.

В сфере IT: есть протоколы передачи данных TCP-IP протоколы нижнего уровня для передачи информации, есть HTTP протокол передачи гипертекста.

Так же относятся стандарты для оборудования возьмем кабель Ethernet (10mb/s-скорость, кабель-толстый, витая пара, оптика)- стандарт 802.3

3. ***Республиканские стандарты*** обязательны для предприятий республиканского и *местного подчинения данной республики* независимо от их ведомственной принадлежности.

Республиканские стандарты обязательны для предприятий республиканского и местного подчинения данной республики независимо от их ведомственной принадлежности.

У нас есть свои стандарты и мы им следуем у себя в республике, не зависимо, что есть гос. Стандарт.

4. ***Стандарты предприятий*** (объединений) *действуют только на предприятии*, утвердившем данный стандарт.

Стандарты предприятий (объединений) распространяются на нормы, правила, методы, составные части изделий и другие объекты, имеющие применение только на данном предприятии; *Стандарты предприятий могут также устанавливать ограничения по применяемой номенклатуре деталей, составных частей, материалов, предусмотренные государственными, отраслевыми или республиканскими стандартами.*

МЕТОДИКА РАЗРАБОТКИ И УТВЕРЖДЕНИЯ СТАНДАРТОВ.

Целесообразность разработки каждого стандарта обосновывается потребностями народного хозяйства и ожидаемым техническим и экономическим эффектом. Для этого *предварительно подбирают и анализируют литературные и производственные данные, устанавливают тенденции развития и перспективные потребности промышленности по стандартизуемым объектам или параметрам.*

Т.е. стандарт разрабатывается тогда, когда изучены потребительские особенности, ожидаемые технические характеристики уже знаем тенденции развития в определенной сфере.

Например, для покупателя телевизора важны размеры экрана, четкость изображения, гарантийный срок, внешний вид и его ремонтопригодность, т.е. возможность быстрого обнаружения повреждений и замены неисправных элементов.

Для завода-изготовителя, кроме указанного, важное значение имеют совершенство конструкции и технологичность составных частей телевизора, определяющих трудоемкость и экономичность его производства

Обязательным этапом является анализ зарубежного опыта и достигнутого там уровня качественных показателей стандартизуемых объектов.

Номенклатура показателей качества должна быть достаточной, чтобы всесторонне и полно характеризовать изделие не только с точки зрения изготовителя, но и с точки зрения потребителя.

ГСС устанавливает шесть стадий разработки стандартов:

1) организация разработки стандарта, составление и утверждение технического задания;
2) разработка проекта стандарта и рассылка его на отзыв;
3) анализ отзывов и разработка окончательной редакции проекта стандарта;
4) подготовка, согласование и представление стандарта на утверждение;
5) рассмотрение, утверждение и регистрация стандарта;
6) издание стандарта и информации о нем.

Контрольные вопросы:

1. Что такое стандартизация?
2. Чем стандартизация отличается от стандарта?
3. Что следует из определения стандартизация? Дайте определение
4. Какая главная цель государственной системы стандартизации?
5. Назовите задачи госстандарта?
6. Назовите и опишите категории стандартов?
7. Назовите 6 стадий разработки стандартов?
8. Какая методика разработки и утверждения стандартов?

Список использованных источников:

1) Орлов, С.А. Технологии разработки программного обеспечения: учебник / С.А. Орлов. – СПб: Питер, 2002. – 464 с.
2) Липаев, В.В. Управление разработкой программных средств: Методы, стандарты, технология / В.В. Липаев. – М.: Финансы и статистика, 1993.
3) Липаев, В.В. Тестирование программ / В.В. Липаев. – М.: Радио и связь, 1986.
4) Липаев, В.В., Технология сборочного программирования / В.В. Липаев, Б.А. Позин, А.А. Штрик. – М.: Радио и связь, 1992.
5) Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством. – М.: Изд-во стандартов, 1990.
6) Лифиц, И.М. Стандартизация, метрология и сертификация /И.М. Лифиц. –М.: Юрайт-издат, 2004. – 335 с.
7) Сертификация сложных технических систем /Л.Н. Александровская [и др.]. – М.: Логос, 2001. – 312 с.
8) Якушев, А.И., Взаимозаменяемость, стандартизация и технические измерения / А.И Якушев, Л.Н. Воронцов, Н.М. Федотов. – М.: Машиностроение, 1986. – 352 с.

ЛЕКЦИЯ 2

Тема: жизненный цикл программных средств. Нормативный документы. Процессы ЖЦ в соответствии со стандартом ISO 12207

Цель: изучить основные процессы продиктованные стандартом ISO 12207

Программное средство – это набор компьютерных программ, процедур и связанных с ними документации и данных.

Это наш инструментарий с его мануалами(документацией)

Жизненный цикл (ЖЦ) ПС – это период времени, который начинается с момента принятия решения о необходимости создания ПС и заканчивается в момент его полного изъятия из эксплуатации.

Процесс – это совокупность взаимосвязанных действий, преобразующих некоторые входные данные в выходные.

Основной нормативный документ, регламентирующий состав процессов ЖЦ ПС, – международный стандарт **ISO/IES 12207**. Стандарт определяет структуру ЖЦ, содержащую процессы, действия и задачи, выполняемые в процессе создания.

Стандарты для ПС:

1. **ISO** (International Organization of Standardization) – Международная организация по стандартизации
2. **IEC** (International Electro-technical Commission) – Международная организация по электротехнике
3. **ISO/IEC 12207** – стандарт, принятый в 1995 году для разработки ПС.

В соответствии со стандартом ISO 12207 все процессы ЖЦ разделены на следующие:

1. Основные процессы;
2. Вспомогательные процессы, обеспечивающие выполнение основных процессов;
3. Организационные процессы.

1) К ОСНОВНЫМ ПРОЦЕССАМ ОТНОСЯТСЯ:

1. **приобретение**;

то когда мы захотели купить какую либо услугу

2. **поставка**;

поставщик услуг следит за работоспособностью своего продукта

3. **разработка**;

разработчик проектирует систему, выбирает средства для разработки, программирует и тестирует модули разработки

4. **эксплуатация;**

проводится тестирование общего ПО

5. **сопровождение**.

Все что касается разработки и поддержки ПО в работоспособном виде или ее усовершенствовании

1) Приобретение *Действия заказчика, приобретающего ПС*:

1.1 инициирование приобретения;

1.2 подготовка заявочных предложений;

1.3 подготовка и корректировка договора;

1.4 надзор за деятельностью поставщика;

1.5 приемка и завершение работы.

2) **Поставка** *-Действия и задачи поставщика, который снабжает заказчика программным продуктом или услугой:*

2.1 инициирование поставки;

2.2 подготовка ответа на заявочное предложение;

2.3 подготовка договора;

2.4 планирование;

2.5 выполнение и контроль;

2.6 проверка и оценка;

2.7 поставка и завершение работы.

3) **Разработка** -*Действия и задачи, выполняемые разработчиком*. Охватывает работы по созданию ПС и его компонентов в соответствии с заданными средствами, *включая оформление проекта и эксплуатационной документации*, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала и т.д.

Разработка включает следующие действия:

3.1 подготовительную работу;

3.2 анализ требований к системе;

3.3 проектирование архитектуры системы;

3.4 анализ требований к ПС;

3.5 проектирование архитектуры ПС;

3.6 детальное проектирование ПС;

3.7 кодирование и тестирование ПС;

3.8 интеграцию ПС;

3.9 квалификационное тестирование;

3.10 интеграцию системы;

3.11 установку ПС;

3.12 приемку ПС.

4) **Эксплуатация** -*Охватывает действия и задачи оператора организации, эксплуатирующей ПС.*

Процесс эксплуатации включает в себя следующие действия:

4.1 подготовительная работа;

4.2 эксплуатационное тестирование;

4.3 эксплуатация системы;

4.4 поддержка пользователей.

5) **Сопровождение**- *Предусматривает действия, выполняемые сопровождающей организацией.* Это внесение *изменений в ПС в целях исправления ошибок*, повышения производительности или адаптации к изменившимся условиям работы. *Изменения, вносимые в ПС, не должны нарушать его целостность.*

Процесс сопровождения включает в себя следующие действия:

5.1 подготовительная работа;
5.2 анализ проблем и запросов на модификацию ПС;
5.3 модификация ПС;
5.4 проверка и приемка;
5.5 перенос ПС в другую среду;
5.6 снятие ПС с эксплуатации.

2) ***К ВСПОМОГАТЕЛЬНЫМ ПРОЦЕССАМ ОТНОСЯТСЯ:***

1) документирование;
2) управление конфигурацией;
3) обеспечение качества;
4) верификация;
5) аттестация;
6) совместная оценка;
7) аудит;
8) разрешение проблем.

3) ***ОРГАНИЗАЦИОННЫЕ ПРОЦЕССЫ:***

1) управление;
2) создание инфраструктуры;
3) усовершенствование;
4) обучение.

СТАДИИ ЖИЗНЕННОГО ЦИКЛА ИНФОРМАЦИОННОЙ СИСТЕМЫ:

1)Предпроектное обследование:

1. ·сбор материалов для проектирования, при этом выделяют формулирование требований, с изучения объекта автоматизации, даются предварительные выводы пред проектного варианта ИС;
2. анализ материалов и разработка документации, обязательно дается технико экономическое обоснование с техническим заданием на проектирование ИС.

2) Проектирование:

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

2.1 предварительное проектирование;

1. · выбор проектных решений по аспектам разработки ИС;
2. · описание реальных компонент ИС;
3. · оформление и утверждение технического проекта (ТП).

2.2 детальное проектирование:

1. · выбор или разработка математических методов или алгоритмов программ;
2. · корректировка структур БД;
3. · создание документации на доставку и установку программных продуктов;
4. · выбор комплекса технических средств с документацией на ее установку.

2.3 разработка техно-рабочего проекта ИС (ТРП).

2.4 разработка методологии реализации функций управления с помощью ИС и описанием регламента действий аппарата управления.

3) Разработка ИС:

1. · получение и установка технических и программных средств;
2. · тестирование и доводка программного комплекса;
3. · разработка инструкций по эксплуатации программно-технических средств.

4) Ввод ИС в эксплуатацию:

1. · ввод технических средств;
2. · ввод программных средств;
3. · обучение и сертификация персонала;
4. · опытная эксплуатация;
5. · сдача и подписание актов приемки-сдачи работ.

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

5) Эксплуатация ИС:

1. · повседневная эксплуатация;
2. · общее сопровождение всего проекта.

После того ,как мы сдали наше ПО в эксплуатацию идет сопровождение и поддержка

Контрольные вопросы:

9. Что такое ЖЦ ПС?
10. Что такое программное средство?
11. Что такое процесс?
12. Стандарты для ПС?
13. В соответствии со стандартном процессы ЖЦ разделены
14. Что относится к основным процессам?
15. Что относится к вспомогательным процессам?
16. Что относится к организационным процессам?
17. Что такое сопровождение?

Список использованных источников:

9) Орлов, С.А. Технологии разработки программного обеспечения: учебник / С.А. Орлов. – СПб: Питер, 2002. – 464 с.

10) Липаев, В.В. Управление разработкой программных средств: Методы, стандарты, технология / В.В. Липаев. – М.: Финансы и статистика, 1993.

11) Липаев, В.В. Тестирование программ / В.В. Липаев. – М.: Радио и связь, 1986.

12) Липаев, В.В., Технология сборочного программирования / В.В. Липаев, Б.А. Позин, А.А. Штрик. – М.: Радио и связь, 1992.

13) Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством. – М.: Изд-во стандартов, 1990.

14) Лифиц, И.М. Стандартизация, метрология и сертификация /И.М. Лифиц. –М.: Юрайт-издат, 2004. – 335 с.

15) Сертификация сложных технических систем /Л.Н. Александровская [и др.]. – М.: Логос, 2001. – 312 с.

16) Якушев, А.И., Взаимозаменяемость, стандартизация и технические измерения / А.И Якушев, Л.Н. Воронцов, Н.М. Федотов. – М.: Машиностроение, 1986. – 352 с.

ЛЕКЦИЯ 3

Тема: Модели ЖЦ в соответствии со стандартом ISO 12207 и 9004:2000. Каскадная модель.

Цель: изучить основные модели продиктованные стандартом ISO 12207. ISO 9004:2000

Модели жизненного цикла ПС

Под моделью ЖЦ ПС понимается структура, определяющая последовательность выполнения и взаимосвязь процессов, действий и задач на протяжении ЖЦ.

соответствуют, регламентированным и детализированным в ***стандартах ISO 9001:2000, ISO 12207*** *основных компонентах профиля стандартов жизненного цикла сложных ПС.*

Модель ЖЦ продиктована стандартом ISO 12207 и зависит от:

1. спецификации;
2. масштабов;
3. условий.

Модель ЖЦ определяет характер процессов его создания, который представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в стадии работ, выполнение которых необходимо и достаточно для создания ПС, соответствующего заданным требованиям.

Модель ЖЦ определяет все стадии от идеи до поддержки и снятия с эксплуатации ПП., описывает какие действия должны происходить на каждом этапе, какой документацией должно сопровождаться разработка.

Модель жизненного цикла АИС— это структура, описывающая процессы, действия и задачи, которые осуществляются и ходе разработки, функционирования и сопровождения в течение всего жизненного цикла системы.

• По возможности нужно выбирать стратегию с наименьшим количеством всевозможных рисков, неопределенных и неконтролируемых событий.

Запомните, что нужно использовать не те методы, которые могут работать, а которые действительно будут работать.

• Если проект связан с рисками, подготовьте несколько запасных стратегий на случай, когда первоначальные методы себя не оправдают

Выбор модели жизненного цикла зависит от специфики, масштаба, сложности проекта и набора условий, в которых АИС создается и функционирует.

Модель ЖЦ АИС включает:

• стадии;

• результаты выполнения работ на каждой стадии;

• ключевые события или точки завершения работ и принятия решений.

В соответствии с известными моделями ЖЦ ПО определяют модели ЖЦ АИС — каскадную, итерационную, спиральную.

Модели жизненного цикла информационной системы:

1. **каскадная модель** 1970-80гг
2. **инкрементная модель**

3. **спиральная модель** 1986г
4. **Компонентно-ориентированная модель**

КАСКАДНАЯ МОДЕЛЬ ЖЦ

В модели водопада, называемой также "каскадная модель жизненного цикла" или "каскадная модель жизненного цикла с обратными связями", сопровождение ПО выделяется в отдельную фазу жизненного цикла.

Каскадная стратегия (однократный проход, водопадная или классическая модель) подразумевает линейную последовательность выполнения стадий создания информационной системы (рис.3.1). Другими словами, переход с одной стадии на следующую происходит только после того, как будет полностью завершена работа на текущей.

Данная модель применяется при разработке информационных систем, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования.

Рис.3.1. Каскадная стратегия

На первом этапе проводится исследование проблемы, которая должна быть решена, четко формулируются все требования заказчика. Результатом, получаемым на данном этапе, является техническое задание (задание на разработку), согласованное со всеми заинтересованными сторонами.

Ставим цели, задачи, исследуем предметную область, тесно общаемся с клиентом, что должен выполнять ПП

На втором этапе разрабатываются проектные решения, удовлетворяющие требованиям, сформулированным в техническом задании. Результатом, получаемым на данном этапе, является комплект проектной документации, содержащей необходимые данные для реализации проекта.

Составляем техническую документацию по проекту исходя из поставленных целей и задач

Третий этап — реализация проекта. Здесь осуществляется разработка программного обеспечения (кодирование) в соответствии с проектными решениями, полученными на втором этапе. Методы, используемые для реализации, не имеют принципиального значения. Результатом, получаемым на данном этапе, является готовый программный продукт.

На четвертом этапе проводится проверка полученного программного обеспечения на предмет соответствия требованиям, заявленным в техническом задании. Опытная эксплуатация позволяет выявить различного рода скрытые недостатки, проявляющиеся в реальных условиях работы информационной системы.

Тестирование блоков отдельных частей ПО

Пятый этап — сдача готового проекта. Главная задача этого этапа — убедить заказчика, что все его требования выполнены в полной мере. **ISO 14764-сопровождение**

Достоинства модели:

- на каждой стадии формируется законченный набор документации, программного и аппаратного обеспечения, отвечающий критериям полноты и согласованности;

- выполняемые в четкой последовательности стадии позволяют уверенно планировать сроки выполнения работ и соответствующие ресурсы (денежные, материальные и людские).

Недостатки модели:

1. - реальный процесс разработки информационной системы редко полностью укладывается в такую жесткую схему. Особенно это относится к разработке нетиповых и новаторских систем;

2. основана на точной формулировке исходных требований к информационной системе. Реально в начале проекта требования заказчика определены лишь частично;

3. основной недостаток – результаты разработки доступны заказчику только в конце проекта. В случае неточного изложения требований или их изменения в течение длительного периода создания ИС заказчик получает систему, не удовлетворяющую его потребностям.

4. существенная задержка в получении результатов;

Задержка в получении результатов проявляется в том, что при последовательном подходе к разработке согласование результатов производится только после завершения очередного этапа работ.

В результате может оказаться, что разрабатываемая АИС не соответствует требованиям, и такие несоответствия могут возникать на любом этапе разработки; кроме того, ошибки могут непреднамеренно вноситься и проектировщиками-аналитиками, и программистами, так как они не обязаны хорошо разбираться в тех предметных областях, для которых разрабатывается АИС.

5. ошибки и недоработки на любом из этапов проявляются, как правило, на последующих этапах работ, что приводит к необходимости возврата;

Возврат на более ранние стадии. ошибки, допущенные на более ранних этапах, обнаруживаются только на последующих стадиях. В результате проект возвращается на предыдущий этап, перерабатывается и только затем передается в последующую работу.

Это может послужить причиной срыва графика и усложнения взаимоотношений между группами разработчиков, выполняющих отдельные этапы.

6. сложность параллельного ведения работ по проекту;

Чем сильнее взаимосвязь отдельных частей проекта, тем чаще и тщательнее должна выполняться синхронизация, тем сильнее зависят друг от друга группы разработчиков. В результате преимущества параллельного проведения работ просто теряются; отсутствие параллелизма негативно сказывается и на организации работы всего коллектива.

7. чрезмерная информационная перенасыщенность каждого из этапов;

Проблема информационной перенасыщенности возникает вследствие сильной зависимости между различными группами разработчиков. Дело в том, что при внесении изменений в одну из частей проекта, необходимо оповещать тех разработчиков, которые использовали (могли использовать) ее в своей работе.

При наличии большого числа взаимосвязанных подсистем синхронизация внутренней документации становится отдельной важнейшей задачей: разработчики должны постоянно знакомятся с изменениями и оценивать, как скажутся эти изменения на полученных результатах.

8. сложность управления проектом;

Сложность управления проектом в основном обусловлена строгой последовательностью стадий разработки и наличием сложных взаимосвязей между различными частями проекта. Регламентированная последовательность работ приводит к тому, что одни группы разработчиков должны ожидать результатов работы других команд, поэтому требуется

административное вмешательство для согласования сроков и состава передаваемой документации.

9. высокий уровень риска и ненадежность инвестиций.

Высокий уровень риска. Чем сложнее проект, тем дольше длится каждый этап разработки и тем сложнее взаимосвязи между отдельными частями проекта, количество которых также увеличивается. Причем результаты разработки можно реально увидеть и оценить лишь на этапе тестирования, т. е. после завершения анализа, проектирования и разработки — этапов, выполнение которых требует значительного времени и средств.

10. Возникновение конфликтов между разработчиками.

Возврат части проекта на ранние стадии обусловлен писком виновных в ошибке. И как следствие ценится не тот руководитель, который имеет высокую квалификацию и большой опыт, а тот, кто может отстоять своих подчиненных.

Контрольные вопросы:

1. Что такое модель ЖЦ АИС?
2. Отчего зависит выбор модели ЖЦ?
3. Что включает в себя модель ЖЦ?
4. Что такое каскадная стратегия?
5. Опишите этапы каскадной стратегии
6. На какие две фазы делится каскадная стратегия?
7. Назовите достоинства и недостатки каскадной стратегии

Список использованных источников:

17) Орлов, С.А. Технологии разработки программного обеспечения: учебник / С.А. Орлов. – СПб: Питер, 2002. – 464 с.

18) Липаев, В.В. Управление разработкой программных средств: Методы, стандарты, технология / В.В. Липаев. – М.: Финансы и статистика, 1993.

19) Липаев, В.В. Тестирование программ / В.В. Липаев. – М.: Радио и связь, 1986.

20) Липаев, В.В., Технология сборочного программирования / В.В. Липаев, Б.А. Позин, А.А. Штрик. – М.: Радио и связь, 1992.

21) Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством. – М.: Изд-во стандартов, 1990.

22) Лифиц, И.М. Стандартизация, метрология и сертификация /И.М. Лифиц. –М.: Юрайт-издат, 2004. – 335 с.

23) Сертификация сложных технических систем /Л.Н. Александровская [и др.]. – М.: Логос, 2001. – 312 с.

24) Якушев, А.И., Взаимозаменяемость, стандартизация и технические измерения / А.И Якушев, Л.Н. Воронцов, Н.М. Федотов. – М.: Машиностроение, 1986. – 352 с.

- https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema3#p32
- http://www.computer-museum.ru/books/n_collection/models.htm

ЛЕКЦИЯ 4

Тема: Инкрементная модель ЖЦ. Процесс разработки АИС по Rad схеме в соответствии со стандартом ISO 12207

Цель: изучить инкрементной модель АИС. Понять принцип работы, в каких случаях ее применяют на практике. Понять достоинства и недостатки данной стратегии

Модели жизненного цикла информационной системы:

5. **каскадная модель** 1970-80гг
6. **инкрементная модель,RAD**
7. **спиральная модель** 1986г
8. **Компонентно-ориентированная модель**

ГОСТ Р ИСО/МЭК ТО 15271–2002 – Информационная технология – Руководство по применению ГОСТ Р ИСО/МЭК 12207 (Процессы жизненного цикла программных средств)

Инкрементная стратегия (англ. increment – увеличение, приращение) подразумевает разработку информационной системы с линейной последовательностью стадий, но в несколько инкрементов (версий), т. е. с запланированным улучшением продукта.

Инкрементная модель является классическим примером инкрементной стратегии разработки ПО, объединяя элементы последовательной водопадной модели с итерационной философией макетирования.

Данная стратегия в реальной жизни выдает проект заказчику от набросками, одну ф-ю написали вылили в продакшин.

методология быстрой разработки приложений инкрементной модели содержит 3 элемента:

1. небольшую команду программистов (от 2 до 10 человек);
2. короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
3. повторяющийся цикл Т.е. выпускаются версии программы по результату общения с клиентом, каждая версия усовершенствуется

Каждая итерация обеспечивает прохождение всех фаз проекта, обеспечивая инкремент (прирост) функциональности. Однако итерация, как правило, недостаточна для выпуска новой версии продукта. По окончании итерации команда разработчиков оценивает и выбирает приоритеты разработки.

Главное рабочий продукт, а не система письменной документации. Т.е. очень скудная тех.документация по проектированию и использованию

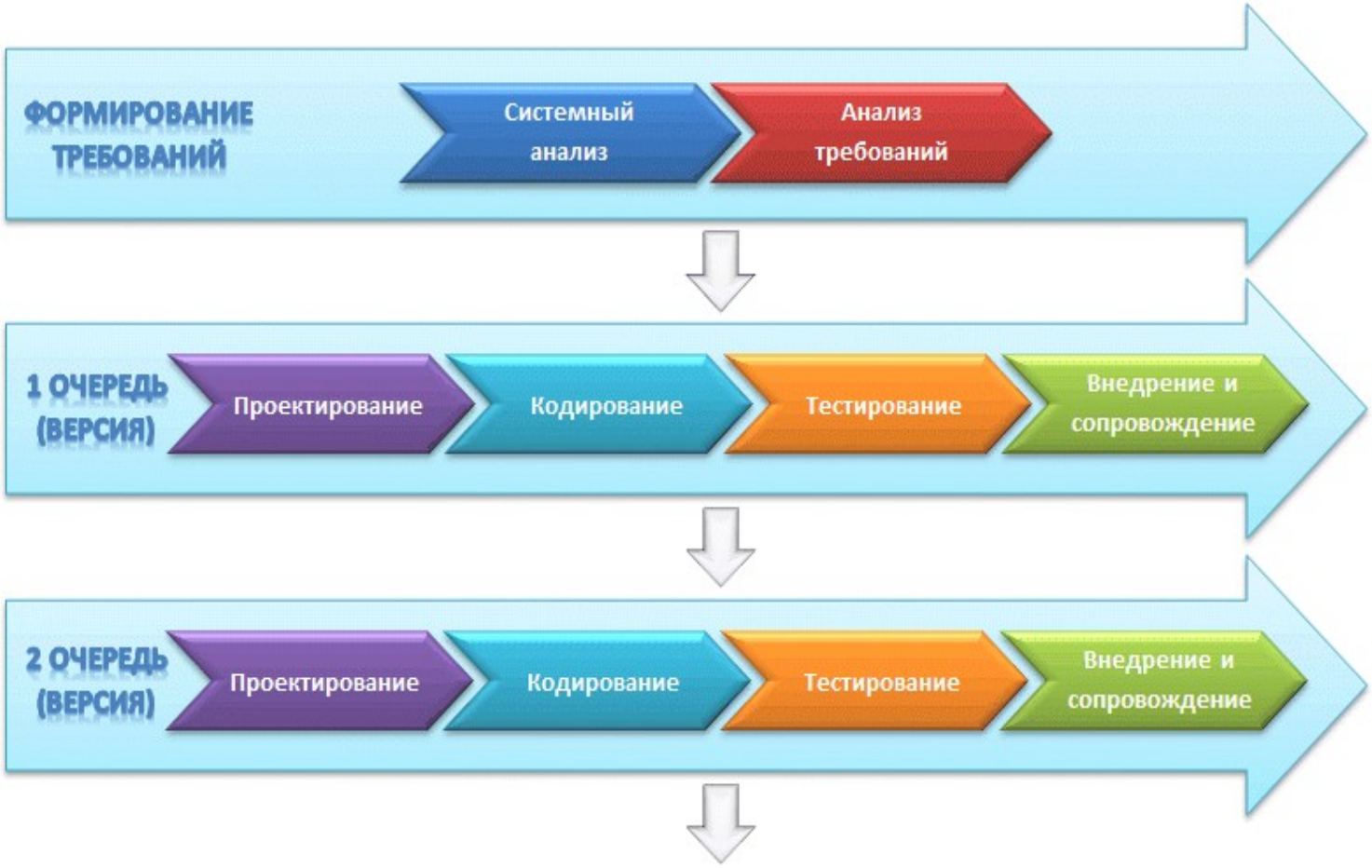


Рис.3.2. Инкрементная стратегия

Принцип работы инкрементной стратегии

В начале работы над проектом определяются все основные требования к системе, после чего выполняется ее разработка в виде последовательности версий. При этом каждая версия является законченным и работоспособным продуктом. Первая версия реализует часть запланированных возможностей, следующая версия реализует дополнительные возможности и т. д., пока не будет получена полная система.

Данная модель жизненного цикла характерна при разработке комплексных систем, для которых имеется четкое видение (как со стороны заказчика, так и со стороны разработчика) того, что собой должен представлять конечный результат (информационная система).

Разработка версиями ведется в силу разного рода причин:

1. -отсутствия у заказчика возможности сразу профинансировать весь дорогостоящий проект;

не достаточно денежных средств

2. - отсутствия у разработчика необходимых ресурсов для реализации сложного проекта в сжатые сроки;

маленький коллектив разработчиков

3. - требований поэтапного внедрения и освоения продукта конечными пользователями. Внедрение всей системы сразу может вызвать у ее пользователей неприятие и только «затормозить» процесс перехода на новые технологии.

Выдают проект частями, так как клиента может не устроить версия ПП

Достоинства и недостатки этой стратегии такие же, как и у классической. Но в отличие от классической стратегии заказчик может раньше увидеть результаты. Уже по результатам разработки и внедрения первой версии он может незначительно изменить требования к разработке, отказаться от нее или предложить разработку более совершенного продукта с заключением нового договора.

Достоинства модели:

- на каждой стадии формируется законченный набор документации;
- выполняемые в четкой последовательности стадии позволяют уверенно планировать сроки выполнения работ и соответствующие ресурсы (денежные, материальные и людские).

Недостатки модели:

1. требования заказчика к информационной системе в начале проекта определены лишь частично;
2. существенная задержка в получении результатов;
3. Возникновение конфликтов между разработчиками.
4. высокий уровень риска и ненадежность инвестиций.
5. сложность управления проектом;
6. чрезмерная информационная перенасыщенность каждого из этапов;
7. сложность параллельного ведения работ по проекту
8. ошибки и недоработки на любом из этапов проявляются, как правило, на последующих этапах работ, что приводит к необходимости возврата;

RAD — Rapid Application Development (быстрая разработка приложений) разновидность инкрементной модели

RAD-модели, приведенный в стандарте ГОСТ Р ИСО/МЭК ТО 15271–2002 и адаптированный под требования стандарта ISO/IEC 12207:1995 (СТБ ИСО/МЭК 12207–2003) [8, 3, 9]. Числами в скобках обозначены номера работ процесса разработки, используемые на соответствующих этапах модели

RAD — Rapid Application Development (быстрая разработка приложений), которая стала основой технологий создания и развертывания программных продуктов.

Данная модель позволяет быстро, качественно и с контролем рисков создать и выпустить проект, отличается от модели инкрементной, тем, что возможен возврат и доработка версии.

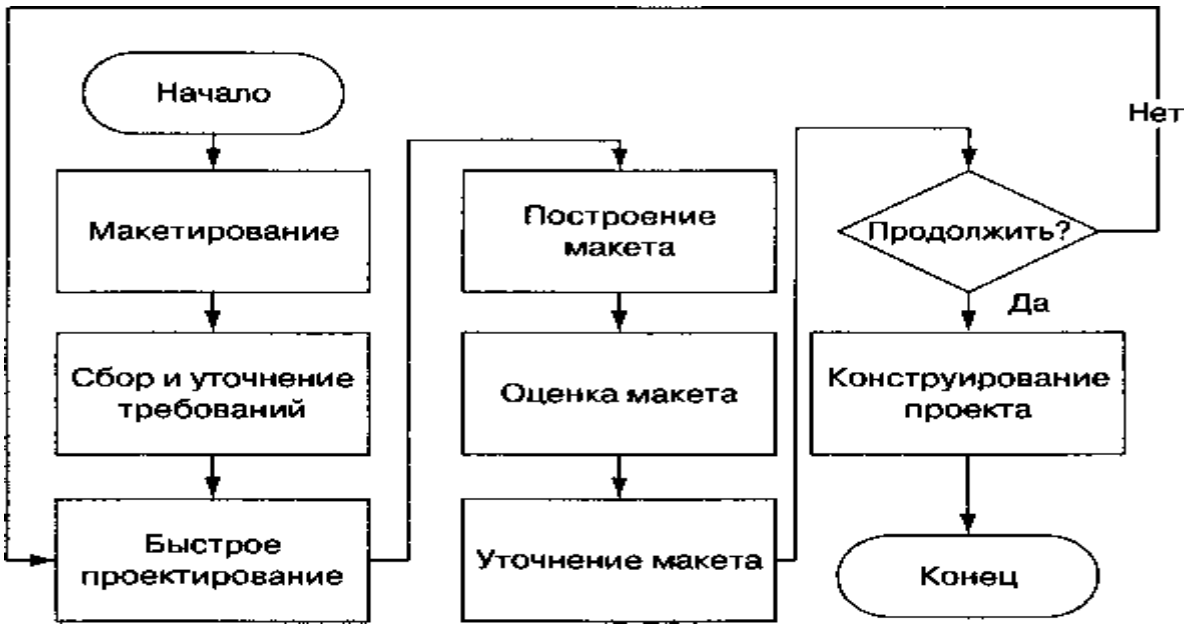


Схема выполнения проектных работ RAD

Макетирование основывается на многократном повторении итераций, в которых участвуют заказчик и разработчик, и начинается со сбора и уточнения требований к создаваемому программному продукту.

Эта модель предусматривает:

1. инструментальную поддержку процесса разработки, минимизацию времени и трудозатрат;

2. использование прототипа для уточнения требований заказчика;

3. цикличность разработки — каждая новая версия продукта основывается на оценке результата работы предыдущей версии заказчиком;

4. постепенное расширение функциональности;

5. распределение ролей в команде разработчика, возможность их совмещения;

6. управление проектом создания программного продукта.

Недостатки Rad модели:

1. Скудный дизайн ПП

2. Отсутствие масштабируемости используется маленькими и средними проектными командами.

3. Нужно выбирать между гибкостью проекта или контролем качества

Случая выбора инкрементной, RAD разработки:

1. для него важна скорость и простота разработки
2. четко определены приоритетные направления разработки проекта
3. разработать приложение нужно в сжатые сроки
4. проект выполняется в условиях ограниченного бюджета
5. главный критерий — интерфейс пользователя
6. есть возможность разбить проект на функциональные компоненты

Контрольные вопросы:

1. что такое экстремальная разработка?
2. Чем отличается инкрементная модель от Rad модели?
3. В каких случаях используют инкрементную и Rad разработку?

4. Принцип работы инкрементной модели
5. Достоинства и недостатки инкрементного проектирования ПП
6. Схема разработки по Rad модели
7. Для чего используют версии проекта?
8. Чем отличается от модели водопада?

Список использованных источников:

- Гагарина, Л.Г. Разработка и эксплуатация автоматизированных информационных систем: учебное пособие для студ. учрежд. СПО/
- Государственный стандарт Российской Федерации. Информационная технология. Сопровождение программных средств
- https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema3#p32
- http://www.computer-museum.ru/books/n_collection/models.htm
- https://worksection.com/blog/rapid-application-development.html

ЛЕКЦИЯ 5

Тема: Спиральная модель в соответствии со стандартом ISO 12207. Процесс разработки, документирования АИС по спиральной схеме. Сравнительный анализ всех моделей

Цель: изучить спиральную модель АИС. Понять принцип работы, в каких случаях ее применяют на практике. Понять достоинства и недостатки данной стратегии

Модели жизненного цикла информационной системы:

9. **каскадная модель** 1970-80гг
10. **инкрементная модель,RAD**
11. **спиральная модель** 1986г

***Следует отметить, что большинство спиральных моделей было разработано ранее принятия международного стандарта* ISO/IEC 12207:1995 [3].** ***В связи с этим необходимо выполнять адаптацию этих моделей с учетом положений действующих национальных аналогов данного стандарта [9] и его новой редакции* ISO/IEC 12207:2008 [4]**

Спиральная стратегия (эволюционная или итерационная модель)подразумевает разработку в виде последовательности версий, но в начале проекта определены не все требования. Требования уточняются в результате разработки версий.

Спиральная модель представляет шаблон процесса разработки ПО, который сочетает идеи итеративной и каскадной моделей.

Суть метода, что весь процесс создания конечного продукта представлен в виде условной плоскости, разбитой на 4 сектора, каждый из которых представляет отдельные этапы его разработки: определение целей, оценка рисков, разработка и тестирование, планирование новой итерации.

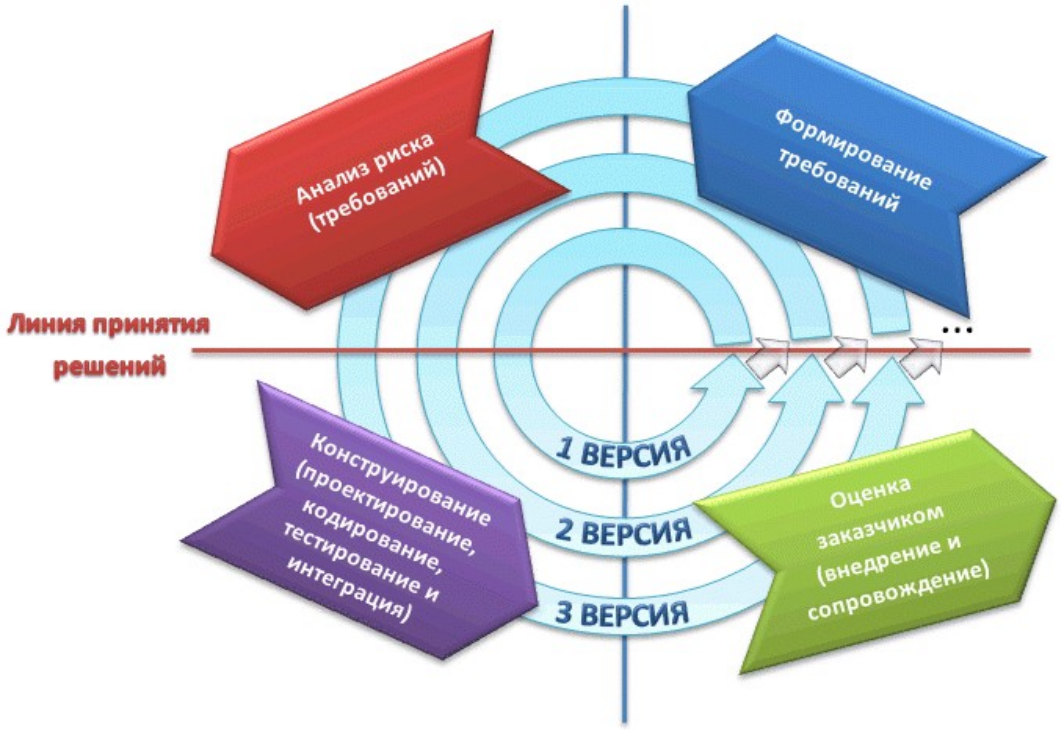
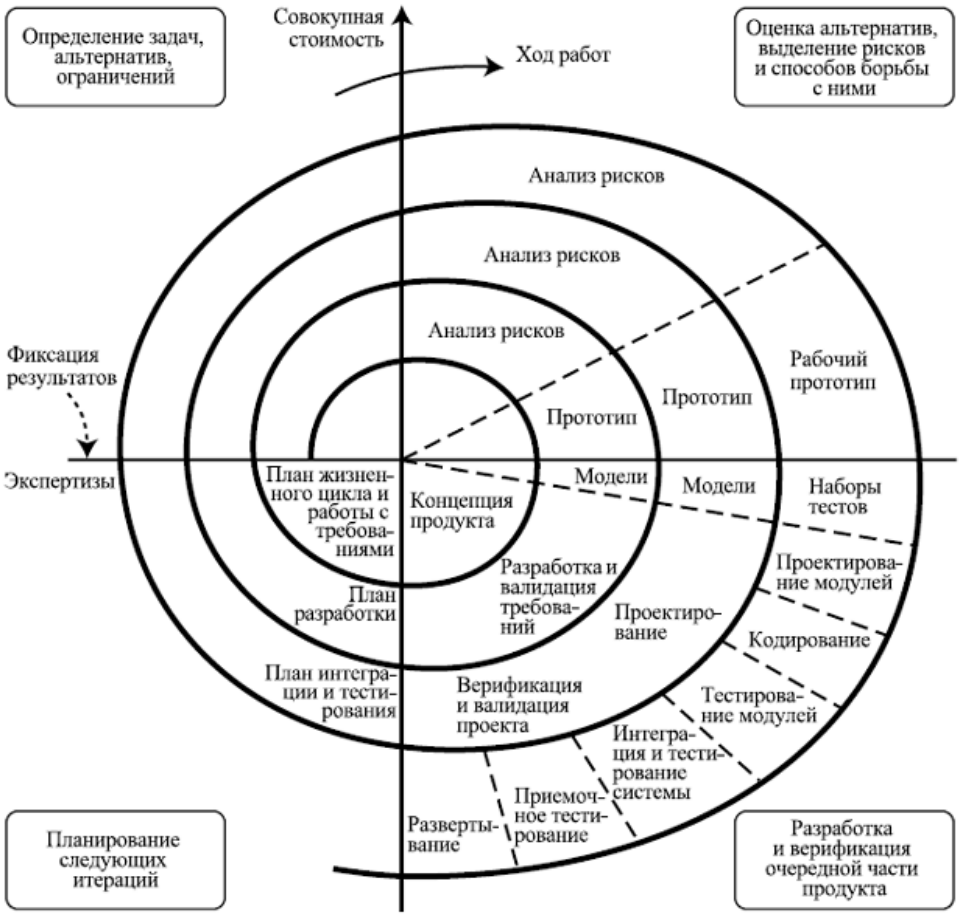


Рис. 3.3. Спиральная стратегия



1. Анализ рисков (**анализ** вероятности того, что произойдут определенные нежелательные события и отрицательно повлияют на достижение целей проекта)
2. валидация требований (подтверждение на основе представления объективных *свидетельств* того, что требования, предназначенные для конкретного использования или применения, выполнены)
3. верификация требований (подтверждение на основе представления объективных свидетельств того, что установленные требования были выполнены

Главная особенность спиральной модели – концентрация на возможных рисках. Для их оценки даже выделена соответствующая стадия.

Основные типы рисков, которые могут возникнуть в процессе разработки ПО:

— Нереалистичный бюджет и сроки;

— Дефицит специалистов;

— Частые изменения требований;

— Чрезмерная оптимизация;

— Низкая производительность системы;

— Несоответствие уровня квалификации специалистов разных отделов.

Принцип работы

Данная модель жизненного цикла характерна при разработке новаторских (нетиповых) систем. В начале работы над проектом у заказчика и разработчика нет четкого видения итогового продукта (требования не могут быть четко определены) или стопроцентной уверенности в успешной реализации проекта (риски очень велики). В связи с этим принимается решение разработки системы по частям с возможностью изменения требований или отказа от ее дальнейшего развития. Как видно из рис.3.3, развитие проекта может быть завершено не только после стадии внедрения, но и после стадии анализа риска.

Достоинства спиральной модели:

1. улучшенный анализ рисков;
2. хорошая документация процесса разработки;
3. гибкость – возможность внесения изменений и добавления новой функциональности даже на относительно поздних этапах;
4. раннее создание рабочих прототипов.

Недостатки спиральной модели

1. может быть достаточно дорогой в использовании;
2. управление рисками требует привлечения высококлассных специалистов;
3. успех процесса в большой степени зависит от стадии анализа рисков;
4. не подходит для небольших проектов.

Когда использовать спиральную модель:

1. когда важен анализ рисков и затрат;
2. крупные долгосрочные проекты с отсутствием четких требований или вероятностью их динамического изменения;
3. при разработке новой линейки продуктов.

Каждая из моделей имеет свои достоинства и недостатки, а также сферы применения в зависимости от специфики разрабатываемой системы, возможностей заказчика и разработчика

Сравнение моделей жизненного цикла

Характеристика проекта	Модель (стратегия)		
	Каскадная	Инкрементная	Спиральная
Новизна разработки и обеспеченность ресурсами	Типовой. Хорошо проработаны технология и методы решения задачи		Нетиповой (новаторский). Нетрадиционный для разработчика
	Ресурсов заказчика и разработчика хватает для реализации проекта в сжатые сроки	Ресурсов заказчика или разработчика не хватает для реализации проекта в сжатые сроки	
Масштаб проекта	Малые и средние проекты	Средние и крупные проекты	Любые проекты
Сроки выполнения проекта	До года	До нескольких лет. Разработка одной версии может занимать срок от нескольких недель до года	
Заключение отдельных договоров на отдельные версии	Заключается один договор. Версия и есть итоговый результат проекта	На отдельную версию или несколько последовательных версий обычно заключается отдельный договор	
Определение основных требований в начале проекта	Да	Да	Нет
Изменение требований по мере развития проекта	Нет	Незначительное	Да
Разработка итерациями (версиями)	Нет	Да	Да
Распространение промежуточного ПО	Нет	Может быть	Да

Контрольные вопросы:

9. суть спиральной модели
10. что такое спиральная модель ПП?
11. Достоинства и недостатки спиральной модели
12. Когда используют спиральную модель
13. Принцип работы спиральной модели
14. На сколько секторов разделена спиральная модель, скажите основные ее возможности
15. Проведите сравнительный анализ всех моделей разработки ПП
16.

Список использованных источников:

- Гагарина, Л.Г. Разработка и эксплуатация автоматизированных информационных систем: учебное пособие для студ. учрежд. СПО/
- Государственный стандарт Российской Федерации. Информационная технология. Сопровождение программных средств
- https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema3#p32
- http://www.computer-museum.ru/books/n_collection/models.htm
- https://worksection.com/blog/rapid-application-development.html
- https://qalight.com.ua/baza-znaniy/spiralnaya-model-spiral-model/

ЛЕКЦИЯ 6

Тема: формирование требований к характеристикам и качеству ПП. Внешнее проектирование.

Цель: изучить внешнее проектирование и понять как устанавливаются требования к разработке

В данной лекции мы поговорим о том какие цели, задачи заказчик должен ставить к проекту, что он должен определить для начала работ по проекту.

Формирование требований к функционированию программных продуктов:

1. - системная *эффективность* целевого применения программных продуктов *определяется степенью удовлетворения потребностей заказчика и пользователей;*

Нравится заказчику –значит эффективно в применении на производстве

2. - каждое *требование должно отражать* отдельно распознаваемую, *измеряемую сущность программного продукта;*

Требование минималичтичный стиль- каждый элемент дизайна должен выполняться в данном тиле. Есть стандарт минимализма и по его пунктам сравниваем насколько хорошо выполнено требование.

3. - для контроля качества и корректности на значимость задач, *следует сопоставлять конкретные требования и сформулированные цели разработки программного продукта*;

Контроль Качество осуществляется по средствам контроля целей, цель- поставлена, цель- выполнена

4. - *для обеспечения качества* программного продукта *необходим набор требуемых характеристик, свойств, их мер* и значений качества для определенных потребителей программного продукта;

Обеспечение Качества мы сравниваем меры и характеристики свойства обьектов со значениями требований

5. следует учитывать, что *требования* обычно *инициируются заказчиком с объемом функций, превышающим тот, который можно реализовать при выделенных ресурсах*;

т.е. заказчик хочет за 100$ полноценный интернет магазин, за 3 дня с расширенным функционалом, а это не соответствует действительности.

6. - *ограниченные ресурсы* для реализации требований функциональной пригодности, могут негативно отражаться на конструктивных характеристиках продукта.

2 разработчика никак не смогут выполнить заказ за 3 дня, качественно и со всем функционалом.

Анализ и разработка требований к ПС(роль заказчика):

Здесь мы определяемся, какую роль заказчик будет выполнять в проекте

1. проекты, *управляемые* пользователем;

заказчик все стадии ЖЦ проекта распределяет и координирует ресурсами сам.

2. проекты, *утверждаемые* пользователем;

заказчик, только документально либо в диалоговой переписке утверждает либо отклоняет решения принятые тимлидером

3. проекты, не зависимые от пользователя.

Все решения принимает руководитель проекта, заказчик только говорит , какие фу-и он хочет видеть в своем ПО.

В процессе разработки требований необходимо решить следующие задачи:

1. Выявить наличие информации, необходимой для выполнения планируемых функций.

Какая информация нам нужна для разработки от заказчика и отдельных отделов разработки

2. Определить трудоемкость и стоимость предстоящей работы.

Рассчитать стоимость работ и сколько понадобиться кадровых единиц для выполнения работ по проекту

3. Обеспечить полноту и точность определения функций, подлежащих выполнению ПС, и их взаимосвязь.

Документируем, какие ф-и на выходе должен выполнять наш продукт, как между собой они будут связаны. *Печать документ не сможет осуществиться, пока не появиться сам документ.*

4. Выявить пространственно-временные ограничения, налагаемые на систему, а также средства системы, которые в будущем могут претерпеть изменения.

Ставим временные рамки для эксплуатации нашего продукта, указываем когда возможно понадобиться обновления или переход к примеру с PHP 7.0 на PHP 7.2

Результатом по выработке требований является соответствующий документ, который должен быть:

1. достаточным для идентификации целей ПС, его среды, преимуществ и недостатков ПС для пользователя, состава и конфигурации ресурсов для его работы;

изучая документ по выработке требований, мы должны точно знать следящее:

-сформулированы точно цели проекта

-выбрана среда для разработки и четко сформулирована область применения ПС

-какие достоинства и недостатки будут при использовании системы

-определена команда для разработки, ее сотом и количество участвующих

2. достаточно полным, чтобы в последующем при разработке исключить серьезные модификации к пересмотру требований;

изучая документ по выработке требований, он настолько должен быть полно описан, *чтобы в средине разработки нам не потребовалось отклоняться от плана за планируемых действий* или *срочно модифицировать продукт, если вышло какое либо обновление*.

Проект должен длительный срок самостоятельно работать без существенных изменений.

3. достаточным для просмотра и утверждения администрацией на основе его реализованности в соответствии с выбранными критериями.

Высшее руководство, когда прочтет документ, должен понять и сравнить реализовались его требования или нет.

Определение целей создания ПС

Цели проекта – это цели, которые должны быть достигнуты в процессе проектирования.

Это то исходя из чего мы решили вообще строить свое проект. Вообще какие цели должны быть достигнуты при проектировании ПП.

Например: повышение эффективности работы, за счёт автоматизации внутренних процессов производства печати документа.

В общем случае цели могут быть сгруппированы в 10 самостоятельных категорий: **ISO 9126**

В каждой из подгрупп мы описываем свои требования, которые мы хоти достичь.

1. **Универсальность**.

(насколько универсально наше ПО оно годится и для печати фото и для печати документов)

2. **Человеческие факторы**.

(должен влиять чел. Фактор на работу системы или это должен быть искусственный интеллект)

3. **Адаптируемость**.

(наше ПО будет работать и на ПК и на смартфонах)

4. **Сопровождаемость**.

(как долго наша фирма будет поддерживать ПО, какой срок, какие услуги предоставлять)

5. **Надежность**. *свойства комплекса программ обеспечивать достаточно низкую вероятность потери работоспособности отказа, в процессе функционирования программного продукта в реальном времени.* ***Основные атрибуты надежности*** *могут быть объективно измерены и сопоставлены с требованиями*

(дает верный результат по заданным критериям, нет скрытых ошибок(битые ссылки или есть в макете функционал, а на программном уровне не реализован))

6. **Безопасность**. **IEC 61508:3** *комплексов программ характеризуется величиной ущерба, возможного при проявлении дестабилизирующих факторов и реализации конкретных угроз – рисков, а также средним временем между проявлениями непредумышленных угроз, нарушающих безопасность.*

(похожа на надежность, но учитываются в безопасности только те угрозы, которые несут материальные потери)
Случайные катастрофический отказ в надежности отразился на безопасности, что в данный момент мы потеряли потенциального зарегистрированного клиента.

7. **Документация**.

(полнота документации и на каких этапах и каким стандартам должна придерживаться)

8. **Стоимость**.

(должны быть указаны расценки на выполнение функционала, чтобы заказчик смог переделится и с его бюджетом и с тем функционалом, который может позволить его бюджет)

9. **Календарный план**.

(устанавливаем сроки сдачи нам результатов по Проекту)

10. **Производительность (эффективность).**

(насколько должна повыситься наша производительность сайта к примеру, измеряем с помощью goole speed**)**

Сформулированные цели ПС, рассматриваемые с точки зрения пользователя, обычно включают следующую информацию:

1. Краткое описание.
2. Определение пользователя.
3. Подробное описание функциональных задач.
4. Документация.
5. Эффективность.
6. Совместимость.
7. Конфигурация.
8. Безопасность. IEC 61508:3
9. Обслуживание.
10. Установка.
11. Надежность.

Внешнее проектирование – процесс описания планируемого поведения разработанного ПС с точки зрения потенциального пользователя.

Внешнее проектирование предназначено для пользователя, что он в первую очередь ожидает от проекта, что хочет получить на выходе.

Описывается, как по его словам должен работать проект. Зашел нажал кнопочку и что в результате должно появится.

К примеру: зашел на сайт сначала, должно появиться окно регистрации/авторизации далее главная страница с информацией сайта и главным меню…

Целью этого процесса является конкретизация внешних взаимодействий будущего ПС без детализации внутреннего устройства.

Получить детальную информацию по бедующему функционалу

Внешний проект представляет собой внешние спецификации ПС, предназначенные для каждой группы специалистов: пользователей и разработчиков.

Для пользователей это будет пользовательская документация, для разработчиков это будет техническая документация.

Предварительный внешний проект содержит основные компоненты разрабатываемого ПС. Т.е из каких главных модулелей будет состоять.

Детальный внешний проект каждой функции пользователя включает:

1. описание входных данных;

это текст, картинки, формулы, числовые данные

2. описание выходных данных;

статистику по какому-либо признаку или шаблон заполненного документа для печати

3. преобразование системы;

расширяемость модулей системы будет возможна или нет, при дополнительном функционале, нам нужно будет нужно переписывать ПО или просто дописать один модуль и подключить его.

4. характеристика надежности;

случайные катастрофические отказы системы, отказы ситуаций, прерывания цикла работоспособности на время, автоматическое обслуживание опасных отказов(рестарт),

5. эффективность; **ISO 9126 :**

5.**1 временная** время отклика программы

5.2 пропускная способность число заданий которые может реализовать на данном ЭВМ в заданном интервале времени

5.3 используемость ресурсов степень загрузки доступных вычислительных ресурсов в течение заданного времени при выполнении функций комплекса программ в установленных условиях

6. замечания по программированию

читабельность кода, стандарты написания кода

Контрольные вопросы:

17. перечислите формирование требований к ПП?
18. Анализ и разработка требований к ПС заказчиком?
19. Какие задачи нужно решить в процессе разработки требований?
20. Что должен одержать в себе документ по выработке т требований?
21. Что такое цель проекта?
22. Назовите 10 категорий целей проекта?
23. Сформулированные цели с точки зрения заказчика, какую содержат информацию?
24. Что такое внешнее проетирование?
25. Что такое внешний проект, на кого рассчитан?
26. Что включает в себя детальный внешний проект?

Список использованных источников:

25) Орлов, С.А. Технологии разработки программного обеспечения: учебник / С.А. Орлов. – СПб: Питер, 2002. – 464 с.

26) Липаев, В.В. Управление разработкой программных средств: Методы, стандарты, технология / В.В. Липаев. – М.: Финансы и статистика, 1993.

27) Липаев, В.В. Тестирование программ / В.В. Липаев. – М.: Радио и связь, 1986.

28) Липаев, В.В., Технология сборочного программирования / В.В. Липаев, Б.А. Позин, А.А. Штрик. – М.: Радио и связь, 1992.

29) Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством. – М.: Изд-во стандартов, 1990.

30) Лифиц, И.М. Стандартизация, метрология и сертификация /И.М. Лифиц. –М.: Юрайт-издат, 2004. – 335 с.

31) Сертификация сложных технических систем /Л.Н. Александровская [и др.]. – М.: Логос, 2001. – 312 с.

32) Якушев, А.И., Взаимозаменяемость, стандартизация и технические измерения / А.И Якушев, Л.Н. Воронцов, Н.М. Федотов. – М.: Машиностроение, 1986. – 352 с.

ЛЕКЦИЯ 7

Тема: проектирование и кодирование программных модулей внешней спецификации.

Цель: изучить внешнее проектирование модулей изучить шаги проектирования и кодирования модулей.

(*Внешнее проектирование модулей, проектирование и кодирование модулей, стиль программирования)*

как создаются модули, как разбиваются на под модули декомпозицию ф-й.

Принцип внешнего проектирования модулей раз*рабатываются внешние взаимосвязи модулей*, которые представляют собой внешнюю спецификацию каждого модуля. *Внешняя спецификация модуля не должна содержать никакой информации о внутреннем устройстве модуля*, об особенностях реализованного в нем алгоритма. Кроме того, недопустимо, чтобы спецификация содержала какие-либо ссылки на вызывающие модули или контексты, в которых этот модуль используется.

Т.е. внешнее проектирование модулей- это более детальный процесс внешнего проектирования.

Внешнее проектирование модуля-включает в себя полное описания внешнего взаимодействия модуля с другими модулями программы(*как передаются данные, в каком формате они должны поступить к другому модулю, что модуль выполняет, какие ф-и до мельчайших подробностей, с какими модулями у него есть внешняя взаимосвязь*.)

Внешняя спецификация модуля содержит следующую информацию:

1. **Имя модуля**.

Указывается имя, с помощью которого можно обратиться к модулю. Для модуля, имеющего несколько входов, составляются отдельные спецификации.

2. **Функцию, назначение модуля (без подробностей**).

Определяется, что делает модуль, когда он вызван, а также его назначение. Этот элемент спецификации не должен содержать сведения о том, как функция реализуется.

3. **Список параметров**.

Определяются число и порядок параметров, передаваемых модулю.

4. **Входные параметры**

Описываются все данные, возвращаемые модулем. Описывается взаимосвязь между входными и выходными данными,
т.е. какие выходные данные на основе каких входных данных получаются. Определяются выходные данные, возвращаемые в вызывающий модуль в случае ошибочных входных данных.

5. **Выходные параметры**

Дается описание всех внешних для программы или системы событий, происходящих при работе модуля, таких, как прием запроса, выдача сообщений об ошибка

После разработки внешних спецификаций модулей приступают к *проектированию* модуля и собственно *программированию* (кодированию) *внутренней логики каждого модуля*.

Этот процесс должен быть тщательно спланирован и состоять из следующих шагов.

Шаги проектирование и кодирование модуля:

1. **Выбор языка программирования.**

Существенное влияние на выбор языка оказывают его возможности обеспечивать надежный процесс получения программ, наличие и специфические особенности компилятора

2. **Проектирование внешних специфик модуля.**

Это процесс определения внешних характеристик каждого модуля.

3. **Проверка правильности внешних спецификаций модуля.**

Правильность спецификаций каждого модуля должна быть проверена сравнением их с информацией о взаимосвязях, полученной при проектировании структуры программы и в результате последующего обсуждения всеми программистами, разрабатывающими вызывающие модули.

4. **Выбор алгоритма и структуры данных.**

К настоящему времени разработано значительное количество алгоритмов и соответствующих структур данных. Следует использовать опыт предыдущих разработок, отчеты, выбрать из имеющихся алгоритмов и структур данных необходимые.

5. **Оформление начала и конца будущего модуля в соответствии с требованиями выбранного языка программирования.**

Предусматривается оформление модуля в соответствии с требованиями принятого языка программирования.

6. **Объявление всех данных, используемых в качестве параметров.**

Записываются соответствующие операторы объявления.

7. **Объявление оставшихся данных.**

Записываются операторы объявления всех оставшихся данных, которые должны быть использованы в модуле.

8. **Детализация логики программы.**

В результате нескольких итераций осуществляется последовательная детализация логики модуля, начиная с достаточно высокого уровня абстракции и заканчивая готовым текстом программы.

На этом шаге используются методы пошаговой детализации и структурного программирования.

9. **Окончательное оформление текста программы.**

Текст модуля проверяется еще раз. При этом вставляются дополнительные комментарии, поясняющие текст программы,

10. **Проверка правильности программы.**

Вручную проверяется правильность модуля - правильность его внутренней логики. Проверка правильности основывается на различных способах чтения текста программы. Проверка

может осуществляться как в форме статического чтения программы, так и в форме динамического чтения.

11. Компиляция модуля.

Этот шаг отмечает переход проектирования к тестированию модуля. Работа над созданием модуля завершена. После компиляции на основе полученной информации проверяется правильность интерпретации компилятором намерений программиста по объявленным данным.

Пошаговая детализация – процесс разложения функции модуля на подфункции. Применяется при декомпозиции модуля.

Структурное программирование – методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Служит для организации проектирования и кодирования программ таким образом, чтобы предотвратить большинство логических ошибок и обнаружить те, которые допущены.

Структурное программирование:

1. Проектирование сверху вниз.
2. Модульное программирование.
3. Структурное кодирование.

Стиль программирования – набор приемов и методов программирования, позволяющих создавать корректные, эффективные и доступные для чтения и понимания программы.

Существует набор рекомендаций:

1. Требование простоты, ясности и удобочитаемости программы.
   Использовать постоянно рефакторин кода, избегать дублирования
2. Использование программистом особенностей языка программирования.
3. Стремление программиста повысить эффективность программы не путем ее улучшения до завершения отладки, а в результате тщательного анализа структур данных, используемых ресурсов и выбора алгоритма реализации.

Контрольные вопросы:

27. Опешите принцип внешнего проектирования модулей?
28. Какую информацию содержат в себе внешние спецификации модулей?
29. Перечислите и опишите шаги проектирования и кодирования модулей?
30. Что такое пошаговая детализация?
31. Что такое структурное программирование?
32. Виды структурного программирования?
33. Что такое Стиль программирования?
34. Назовите 3 рекомендацию по кодированию модулей?

Список использованных источников:

33) Орлов, С.А. Технологии разработки программного обеспечения: учебник / С.А. Орлов. – СПб: Питер, 2002. – 464 с.

34) Липаев, В.В. Управление разработкой программных средств: Методы, стандарты, технология / В.В. Липаев. – М.: Финансы и статистика, 1993.

35) Липаев, В.В. Тестирование программ / В.В. Липаев. – М.: Радио и связь, 1986.

36) Липаев, В.В., Технология сборочного программирования / В.В. Липаев, Б.А. Позин, А.А. Штрик. – М.: Радио и связь, 1992.

37) Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством. – М.: Изд-во стандартов, 1990.

38) Лифиц, И.М. Стандартизация, метрология и сертификация /И.М. Лифиц. –М.: Юрайт-издат, 2004. – 335 с.

39) Сертификация сложных технических систем /Л.Н. Александровская [и др.]. – М.: Логос, 2001. – 312 с.

40) Якушев, А.И., Взаимозаменяемость, стандартизация и технические измерения / А.И Якушев, Л.Н. Воронцов, Н.М. Федотов. – М.: Машиностроение, 1986. – 352 с.

ЛЕКЦИЯ 8

Тема: общие положения о тестировании и отладки программы. Принципы и методы тестирования. Проектирование тестовых наборов.

Цель: изучить общие положения о тестировании и отладки программы

Тестирование – этап ЖЦ (40-60% от всего ЖЦ).

Цель тестирования – процесс многократного выполнения программы с целью обнаружения ошибок. Результатом тестирования являются исходные данные для отладки.

Отладка – процесс, позволяющий получить программу, функционирующую с требующимися характеристиками в заданной области изменения входных данных.

Процесс отладки включает:

1. Действия, направленные на выявление ошибок (тестирование).
2. Диагностику и локализацию ошибок (определение характера и местонахождения ошибок).
3. Внесение исправлений в программу с целью удаления ошибок.

Существуют различные формы тестирования:

1. Доказательство.
2. Контроль.
3. Испытания.
4. Аттестация.

Методы тестирования:

1) **Ручное тестирование-** часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. *Оно проводится тестировщиками или обычными пользователи путем моделирования возможных сценариев действия пользователя.*

2) ***Статическое тестирование*** - *производят с использованием ручных методов* (по данным IBM определяются 30 – 80% ошибок). *Собирается собрание, цель которого -*

обнаружение ошибок, но не их устранение. Процедура статистического тестирования включает инспекцию исходного текста с применением набора правил и приемов обнаружения ошибок. Обычно группа состоит из 4-х человек: руководитель группы (не автор), автор, проектировщик и специалист по тестированию.

2.1 символический – *формальный анализ текста программы на языке программирования.* Операторы и операнды анализируются в символьном виде, поэтому метод иногда называют символическим тестированием.

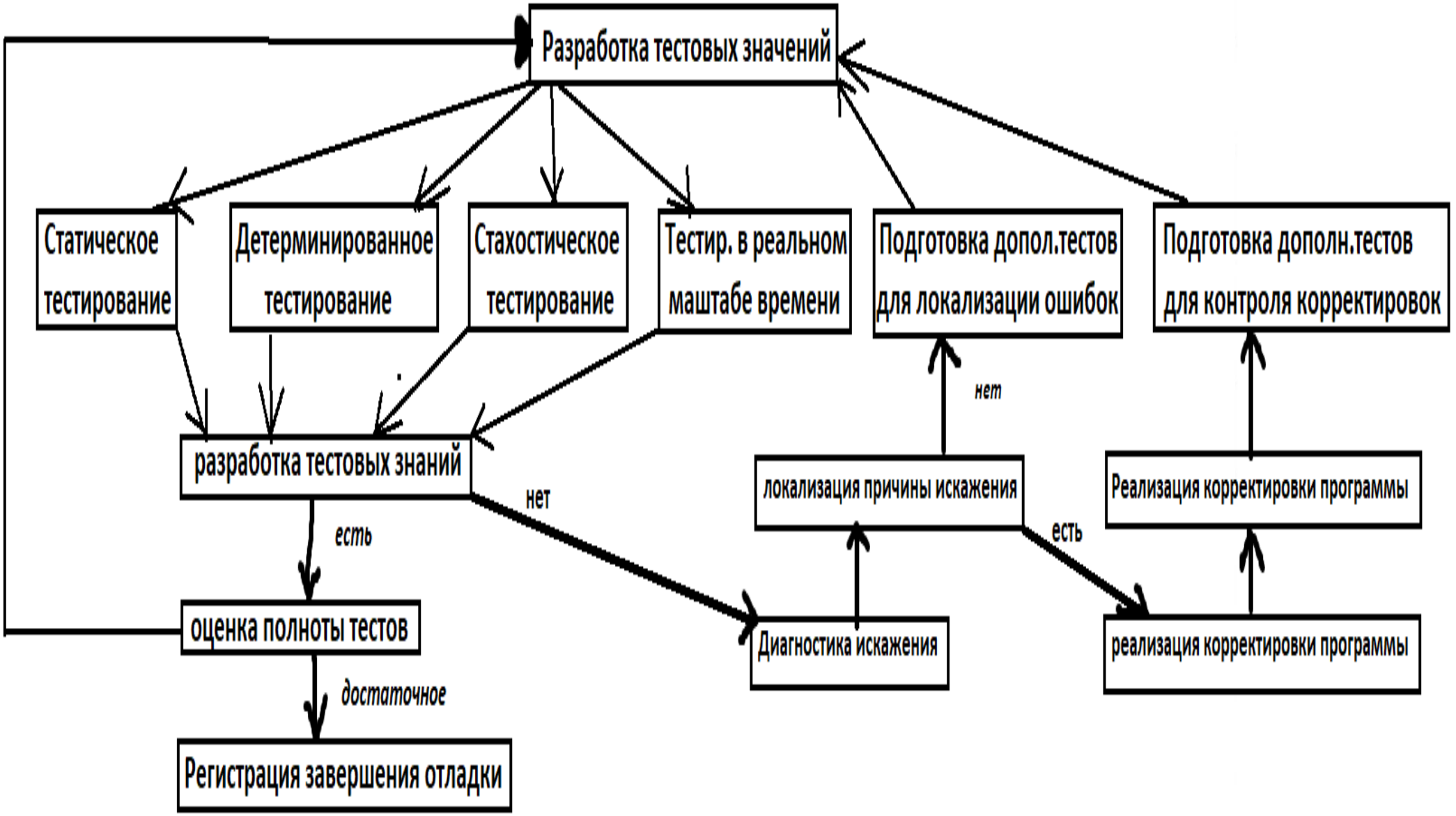
3.1 Детерминированное тестирование – *требует многократного выполнения программы на ЭВМ с использованием специальных тестовых наборов данных.* Контролируется каждая комбинация исходных данных, результаты и каждое утверждение из спецификации программы. Трудоемко, поэтому *применяется для отдельных модулей или небольших ПП*.

3.2 Стохастическое тестирование – *перебрать все варианты невозможно, поэтому используется множество случайных величин с соответствующими распределениями*, для сравнения результатов также используются распределения случайных величин. Применяется для обнаружения ошибок перед детерминированным тестированием. Стохастическое тестирование лучше всего подвержено автоматизации (применяются генераторы случайных чисел).

Методы тестирование

3

РАЗРАБОТКА ТЕСТОВЫХ ЗНАНИЙ


Разработка тестовых значений
Статическое тестирование
Детерминированное тестирование
Стахостическое тестирование
Тестир. в реальном маштабе времени
Подготовка допол.тестов для локализации ошибок
Подготовка дополн.тестов для контроля корректировок
разработка тестовых знаний
нет
есть
оценка полноты тестов
достаточное
Регистрация завершения отладки
нет
локализация причины искажения
есть
Диагностика искажения
Реализация корректировки программы
реализация корректировки программы

ТЕСТИРОВАНИЕ МОДУЛЕЙ

Существует два метода проектирования тестовых наборов данных:

1. ***Стратегия структурного тестирования*** *– тестирование программы как «белого ящика»*, т.е. стратегия тестирования, управляемая логикой программы.
2. ***Стратегия функционального тестирования*** *– тестирование программы как «черного ящика»*, т.е. тестирование по входу и выходу.

СТРУКТУРНОЕ ТЕСТИРОВАНИЕ

Структурное тестирование- Детальное изучение текста программы и построение тестовых наборов входных данных осуществляются на основе одного из четырех.

Критерии построения тестовых наборов:

1) Покрытие операторов.

2) Покрытие узлов ветвлений (решений).

3) Покрытие условий.

4) Комбинаторное покрытие условий.

ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ.

Существует три метода:

1) *Метод эквивалентного разбиения*. - Выделение классов эквивалентности. - Построение тестов.

2) *Метод анализа граничных значений* – исследование ситуаций, возникающих на и вблизи границ эквивалентного разбиения.

3) *Метод функциональных диаграмм*.

• В спецификации программы выделяются причины и следствия. • Каждой причине и следствию присваивается уникальный номер. • Анализируются семантическое содержание спецификации, которое преобразуется в граф, связывающий причины и следствия. • Диаграмма снабжается примечаниями. • По функциональной диаграмме строится таблица решений. • Столбцы решений преобразуются в тесты.

Контрольные вопросы:

35. Что такое тестирование?
36. Какие методы тестирования вы знаете?
37. Какие методы проектирования тестовых наборов знаете?
38. Что такое структурное тестирование?
39. Что такое функциональное тестирование?

Список использованных источников:

41) Орлов, С.А. Технологии разработки программного обеспечения: учебник / С.А. Орлов. – СПб: Питер, 2002. – 464 с.

42) Липаев, В.В. Управление разработкой программных средств: Методы, стандарты, технология / В.В. Липаев. – М.: Финансы и статистика, 1993.
43) Липаев, В.В. Тестирование программ / В.В. Липаев. – М.: Радио и связь, 1986.
44) Липаев, В.В., Технология сборочного программирования / В.В. Липаев, Б.А. Позин, А.А. Штрик. – М.: Радио и связь, 1992.
45) Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством.
46) Лифиц, И.М. Стандартизация, метрология и сертификация /И.М. Лифиц. –М.: Юрайт-издат, 2004. – 335 с.
47) Сертификация сложных технических систем /Л.Н. Александровская [и др.]. – М.: Логос, 2001. – 312 с.
48) Якушев, А.И., Взаимозаменяемость, стандартизация и технические измерения / А.И Якушев, Л.Н. Воронцов, Н.М. Федотов. – М.: Машиностроение,

ЛЕКЦИЯ 9

Тема: методики тестирования. Признаки критического модуля. Отладка ПО. Методы и принципы отладки. Критерии завершения тестирования ГОСТ Р ИСО/МЭК 12119-2000

Цель: изучить общие положения о тестировании и отладки программы. Изучить основные признаки критического модуля

МЕТОДИКА ТЕСТИРОВАНИЯ ПС.

Методики тестирования:

1. *Тестирование элементов* (модулей), проверяющее результаты кодирования ПС.

Проверяют результаты модулей, соответствуют действительности или нет

2. *Тестирование интеграции*, ориентированное на выявление ошибок этапов проектирования ПС.

Проверяют на правильность проектирования ПО, допущены ли были при проектировании системы ошибки.

3. *Тестирование правильности*, проверяющее корректность этапа анализа требований к ПС.

Вводим требования , которые мы получили на этапе анализа и вводим в систему и сравниваем соответствие результаты.

Хотели, чтобы система распознавала лица, сканируем изображение и сморим распознает или нет и какие ошибки выдает.

4. *Системное тестирование*, выявляющее дефекты этапа системного анализа ПС.

Проверяем на несоответствия Программного оборудования. К примеру будет ли работать система при малом объёме оперативной памяти.

Тестирование модулей подвергаются:

1. Интерфейс.
2. Внутренние структуры данных.
3. Независимые пути.
4. Пути обработки ошибок.
5. Граничные условия.

При тестировании путей обнаруживаются следующие категории ошибок:

1. Ошибочные вычисления.
2. Некорректные сравнения.
3. Неправильный поток управления.

Для тестирования модуля необходимы дополнительные средства:

• ***Драйвер*** *– управляющая программа, которая принимает исходные данные и ожидаемые результаты тестовых вариантов*, запускает в работу тестируемый модуль, получает из модуля реальные результаты и формирует донесение о тестировании.

• ***Заглушка*** – замещает модули, которые вызываются тестируемым модулем.

ТЕСТИРОВАНИЕ КОМПЛЕКСА ПРОГРАММ

Цель сборки и тестирования комплекса программ (интеграций) – взять модули, протестированные как элементы, и построить программную структуру, требуемую проектом.

Проверить правильную ли систему мы спроектировали и реализовали зависимости от поставленных задач в начале проекта .

соответствует ли система требования заказчика, все ф-и выполняет или нет.

Два способа комбинирования модулей в систему:

• ***Монолитное тестирование*** *– каждый модуль тестируется независимо друг от друга*, последовательно или параллельно, затем модули собираются в программу за «один раз». **Преимущество**: можно распараллелить процесс, тем самым убыстрить его выполнение.

• ***Пошаговое тестирование*** *– постепенно тестируем* либо нисходящее (можно показать программу сразу), либо восходящее тестирование (система не существует до тех пор, пока не будет добавлен последний модуль)

Постепенно тестируем либо готовое элемент либо тестируем его в процессе разработки.

Признаки критического модуля:

это то когда нужно созданный модуль разделять или исправлять ошибки

1. ***Реализует несколько требований к системе.***

Требование и высокая скорость считывания и обработка файлов, но при этом без использования SSD накопитель и с маленькой ОП.

2. ***Имеет высокий уровень управления.***

Сильно связный с другими модулями, может вывести всю систему сиз работоспособного состояния

3. **Имеет высокую сложность или склонность к ошибкам.**

Не понятно как работает его внутренние ф-и на чем они завязаны или есть вероятность, что при большом количестве вводимых данных наш модуль выйдет из строя.

4. **Имеет определенные требования к производительности обработки. Тестирование функций. Тестирование системы**

Предъявлено много требований одному модулю

5. **Тестирование удобства использования.**

Не удобно его подключать к системе или занимает длительный процесс времени(написание доп.скрипта)

6. **Тестирование на предельных объемах.**

Есть задачи, есть предметная область, должны прогнать модуль на определенном объёме входных данных и отследить, как себя ведет модуль.

7. **Тестирование на предельных нагрузках.**
8. **Удобства эксплуатации.**
9. **Защита**.
10. **Производительность**.
11. **Требования к памяти**.
12. **Конфигурация оборудования**.

Должен иметь определенный набор аппаратного и программного обеспечения и прошивок и сопроводительной документации по их использованию

13. **Совместимость**.
14. **Удобства установки**.
15. **Надежность**.
16. **Восстановление**.
17. **Удобства обслуживания**.
18. **Документация.**

Приемо-сдаточные испытания

Критерии завершения тестирования: ГОСТ Р ИСО/МЭК 12119-2000

1. Выбирается какое-то *количество заранее установленных ошибок или время проверки*.
2. При проведении тестирования *тесты должны стать неудачными*.
3. *По графику*.

Цель Тестирование правильности– *подтвердить, что функции, описанные в спецификации требований, соответствуют ожиданиям заказчика*.

элементов информации, вырабатываемых в процессе разработки ПС:

проверяем каждый пункт заданным требования и его описанию, есть он в наличии или нет, если нет, то добавляем его.

1. Спецификация требований к ПС.
2. Руководство пользователя.
3. Системная спецификация.

Представление требований системе

4. План программного проекта.
5. Спецификация проектирования.
6. Листинги исходных текстов программ.
7. План и методика тестирования.
8. Руководство по работе и инсталляции.
9. .exe код выполняемой программы.
10. Описание БД.
11. Руководство пользователя по настройке.
12. Документ сопровождения.
13. Стандарты и методики разработки программного средства.

ОТЛАДКА

Отладка -тап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

Чтобы понять, где возникла ошибка, приходится: узнавать текущие значения переменных; выяснять, по какому пути выполнялась программа

Отладка – совокупность действий:

1. Разработка тестовых данных.
2. Динамическое и статическое тестирование.

3. Диагностика, локализация причин отклонения, корректировки, изменения.

Два исхода отладки:

• Причина ошибки найдена, уничтожена.

• Не найдена.

Методы отладки

1. ***Аналитические*** – используют методы дедукции и индукции.
2. ***Экспериментальные*** – методы «грубой силы»: трассировка, отладочная печать, трассировка переменных, потоков управления.

Принципы отладки:

1. Использовать средства отладки как вспомогательные.
2. Избегать экспериментирования.
3. Думать.
4. Остерегаться ошибок при процессе корректировки.
5. Все исправления следует документировать.

Контрольные вопросы:

40. Что такое отладка?
41. Какую совокупность действий включает в себя отладка?
42. Методы отладки?
43. Какие исходы отладки вы знаете?
44. Какие элементы информации вырабатываются в процессе разработки ПС?
45. Какие критерии завершения тестирования вы знаете?
46. Назовите признаки критического модуля?
47. Назовите методики тестирования?

Что подвергается тестированию модулей?

Список использованных источников:

49) Орлов, С.А. Технологии разработки программного обеспечения: учебник / С.А. Орлов. – СПб: Питер, 2002. – 464 с.

50) Липаев, В.В. Управление разработкой программных средств: Методы, стандарты, технология / В.В. Липаев. – М.: Финансы и статистика, 1993.

51) Липаев, В.В. Тестирование программ / В.В. Липаев. – М.: Радио и связь, 1986.

52) Липаев, В.В., Технология сборочного программирования / В.В. Липаев, Б.А. Позин, А.А. Штрик. – М.: Радио и связь, 1992.

53) Сертификация продукции. Международные стандарты и руководства ИСО/МЭК в области сертификации и управления качеством.

54) Лифиц, И.М. Стандартизация, метрология и сертификация /И.М. Лифиц. –М.: Юрайт-издат, 2004. – 335 с.

55) Сертификация сложных технических систем /Л.Н. Александровская [и др.]. – М.: Логос, 2001. – 312 с.

56) Якушев, А.И., Взаимозаменяемость, стандартизация и технические измерения / А.И Якушев, Л.Н. Воронцов, Н.М. Федотов. – М.: Машиностроение,

ЛЕКЦИЯ 10

Тема: тестирование документации. Структура мануалов. IEEE Standard SRS Template (спецификации требования к программному обеспечению) Введение.

Цель: понять основные методы и принцип тестирования документации изучить спецификацию требований к ПО

Виды тестирования:

ТЕСТИРОВАНИЕ ДОКУМЕНТАЦИИ

не все организации уделяют достаточное количество времени разработке стоящей документации… Очень часто нам не счастливится иметь дело с толковым программным продуктом и невзрачным, непонятным и беспомощным документным сопровождением.

Техническая документация–набор документов, используемых при проектировании (конструировании), создании (изготовлении) и использовании (эксплуатации) программного и аппаратного обеспечения.

В целом, документация создается для возможности грамотно и без паники найти выход или решение из любой возникшей проблемной ситуации человеку из самым низким знанием принципов разработки программного обеспечения. От этого принципа необходимо отталкиваться, продумывая содержание и структуру наших мануалов.

Структура мануалов:

1. ***Полнота и соответствие действительности***. Любая документация должна содержать описание именно той функциональности, которая присутствует в приложении. И данное описание должно касаться абсолютно всей функциональности, а не только наиболее значимой.

Каждая функция, которая прорабатывается на системе должна быть описана, как правильно ее выполнить, в плоть до нажатии скачать документ

Мы должны проверить все заявленные разработчиком ф-и программы.

2. ***Навигация***. И не просто навигация, а удобная навигация. У пользователя никогда не должно возникать проблем с поиском необходимой ему информации. Все деревья файлов, закладки и прочее должны быть на видном месте сразу, как пользователь открывает документ. Алфавитный указатель, поиск – должно присутствовать все, что поможет найти решение или описание проблемы.

Это основные разделы сайта

Если сайт-хранилище данных , то должно быть максимально расширенный функционал поиска, алфавитный, страничный, по авторам, по названию, по новизне…

И это должно быть максимально понятно

3. От пункта выше, вытекает ***структурированность документации***. Все документы должны находится в полном порядке, по разделах. Текст должен быть также с четкой структурой – чтобы можно было в любой момент вспомнить, где остановился или понять, в каком абзаце содержится именно та информация, которая нам необходима.

4. ***Инструкции должны присутствовать везде***. Даже при выполнении абсолютно одинаковых манипуляций с программой – необходимо пошаговое описание действий во всех случаях. Это может быть, как и прямое повторение инструкций, так и ссылка на уже существующие.

Одинаковые действия тоже документируются или ссылаются на уже описанные действия.

5. ***Термины и их значение***. В любой документации может использоваться масса терминов, аббревиатур и сокращений. Каждый из этих сущностей должно иметь свое значение и расшифровку.

6. ***Доступность пользователю***. Документация должна быть максимально понятной для любой целевой аудитории.

В пользовательской документации не должно быть реплик от разработчиков, их специфике разговора.

7. ***документация с учетом иностранных пользователей*** – необходимо привлечение специалистов данного лингвистического сектора, вплоть до носителей языка.

Программный документ — документ, содержащий в зависимости от назначения данные, необходимые для разработки, производства, эксплуатации, сопровождения программы или программного средства

основных типа документации на ПО:

1. архитектурная/проектная — обзор программного обеспечения, включающий описание рабочей среды и принципов, которые должны быть использованы при создании ПО

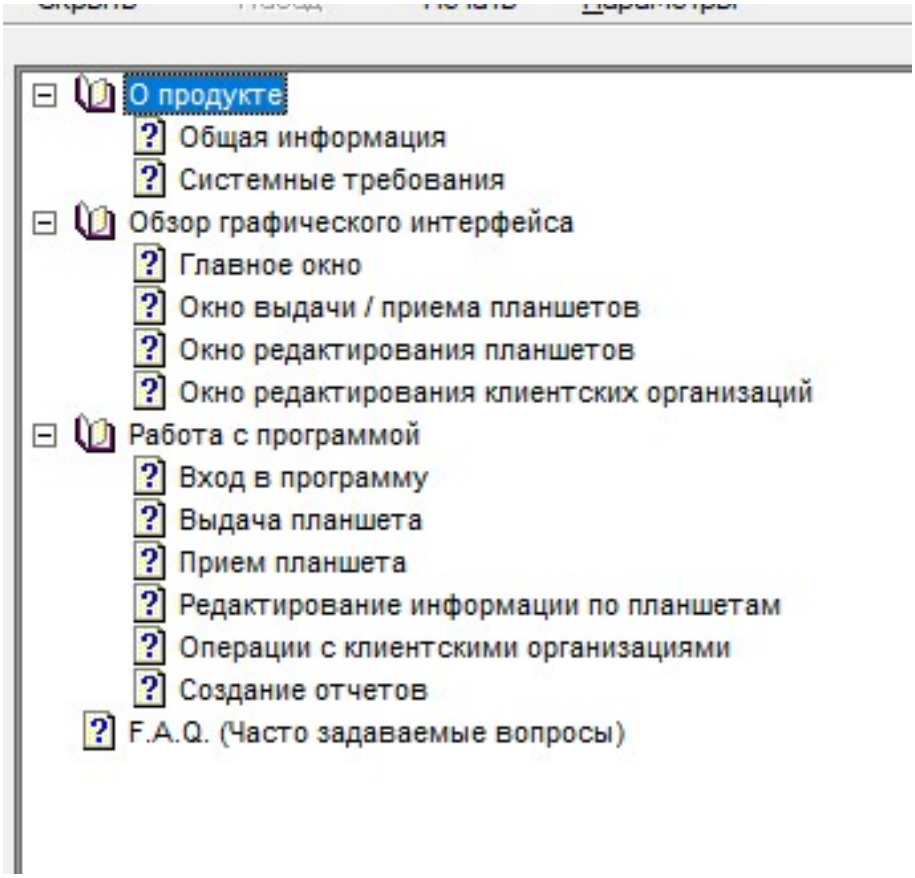
какое ПО и среды будут использоваться при создании определённого проекта

2. техническая — документация на код, алгоритмы, интерфейсы,API

документация для разработчиков. Возможность читать документацию по проекту для сопровождения и разработки, чужим разработчиком

3.пользовательская — руководства для конечных пользователей, администраторов системы и другого персонала

документация для персонала, пользователей



СПЕЦИФИКАЦИЯ ТРЕБОВАНИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

IEEE Standard SRS Template

***Software Requirements Specification*, SRS**

В этом разделе обсуждается каждая основная часть SRS. Эти части упорядочены ниже в виде плана, который может использоваться как образец при написании SRS.

Хотя SRS не обязана в точности следовать этому плану или использовать такие же заголовки частей, хорошая SRS должна включать в себя всю приведенную информацию.

Содержание SRS:

1. Введение
    1. Назначение
    2. Область применения
    3. Определения, акронимы и сокращения
    4. Ссылки
    5. Обзор
2. Общее описание
    1. Позиционирование продукта
    2. Функции продукта
    3. Пользовательские характеристики
    4. Ограничения
    5. Предположения и зависимости
3. Специфические требования
4. Приложения

5. Индекс

. Назначение (1.1 SRS)

В этом подразделе следует:

1. Определить назначение SRS;

2. Задать целевую аудиторию SRS.

Область применения (1.2 SRS)

Этот подраздел должен:

1. Идентифицировать производимый продукт по имени (например, Host DBMS, Report Generator и т.д.);

2. Пояснять, что должен делать программный продукт, а также, при необходимости, чего он не должен делать;

3. Описать применение программного обеспечения, включая выгоды, намерения и цели;

4. Согласовываться со сходными положениями спецификаций верхнего уровня (например, спецификацией требований к системе), если они существуют.

Определения, акронимы и сокращения (1.3 SRS)

Этот подраздел должен представлять определения всех терминов, акронимов и сокращений, необходимых для правильной интерпретации SRS. Эта информация может быть представлена в виде ссылок на одно или более приложений к SRS либо на другие документы.

Ссылки (1.4 SRS)

Данный подраздел должен:

1. Представлять полный перечень документов, на которые есть ссылки где-либо в SRS;

2. Идентифицировать каждый документ по названию, отчетному номеру (если применимо), дате и опубликовавшей организации;

3. Задавать источники, из которых могут быть получены документы, на которые имеются ссылки.

Эта информация может быть представлена в виде ссылки на приложение или другой документ.

Обзор (1.5 SRS)

Данный подраздел должен:

1. Описывать содержимое остальной части SRS;

2. Пояснять организацию SRS.

Контрольные вопросы:

18. что такое техническая документация?
19. Структура мануалов
20. Основные типы документации на ПО?
21. Спецификации требований программного обеспечения
22. Что включает в себя раздел введние?

Список использованных источников:

1. Государственный стандарт Российской Федерации. Информационная технология. Сопровождение программных средств
2. Основы маркетинга учебное пособие Суркова Е.В.

3. https://www.drexplain.ru/help/getting_started.php?ms=AAAA&st=MA%3D%3D&sct=MA%3D%3D&mw=MjUw
4. https://www.elma-bpm.ru/kb/help/Platform/content/Configs_Vkladka_Menu_index.html?ms=AgAAAAAAAAAAAAAAAAAQ%3D%3D&st=MA%3D%3D&sct=MA%3D%3D&mw=MjQw

ЛЕКЦИЯ 11

Тема: IEEE Standard SRS Template (спецификации требования к программному обеспечению) Общее описание(позиционирование продукта, функции продукта, пользовательские характеристики, ограничения, предположения и зависимости)

Цель: изучить спецификацию требований к ПО.

Содержание SRS:

6. Введение
    1. Назначение
    2. Область применения
    3. Определения, акронимы и сокращения
    4. Ссылки
    5. Обзор
7. Общее описание
    1. Позиционирование продукта
    2. Функции продукта
    3. Пользовательские характеристики
    4. Ограничения
    5. Предположения и зависимости
8. Специфические требования
9. Приложения
10. Индекс

ОБЩЕЕ ОПИСАНИЕ (РАЗДЕЛ 2 SRS)

Этот раздел SRS должен описывать общие факторы, оказывающие влияние на продукт и требования к нему. В этом разделе не приводятся специфические требования. В нем подготавливается основа для требований, которые детально определяются в Разделе 3 SRS, и приводится информация, облегчающая их понимание.

Этот раздел обычно состоит из шести подразделов:

1. Позиционирование продукта;
2. Функции продукта;
3. Пользовательские характеристики;
4. Ограничения;
5. Предположения и зависимости;
6. Распределение требований.

Позиционирование продукта (2.1 SRS)

Этот подраздел позиционирует продукт среди других связанных продуктов. Если продукт является независимым и полностью самодостаточным, это следует отразить здесь. Если SRS определяет продукт, который является компонентом большей системы, как это часто бывает, то данный подраздел должен связать требования к большей системе с функциональностью программного обеспечения и определить интерфейсы между системой и программным обеспечением.

Могут быть полезными блок-схемы, показывающие основные компоненты большей системы, связи между ними и внешние интерфейсы.

Этот подраздел должен также описывать, как программное обеспечение работает под действием различных ограничений. Например, эти ограничения могут включать:

1. Системные интерфейсы;
2. Пользовательские интерфейсы;
3. Аппаратные интерфейсы;
4. Программные интерфейсы;
5. Коммуникационные интерфейсы;
6. Память;
7. Операции;
8. Требования к адаптируемости на месте.

2*.1.1. Системные интерфейсы*

Следует перечислить все системные интерфейсы и идентифицировать функциональность программного обеспечения для выполнения требований к системе, а также описание интерфейса для соответствия системе.

2.1.2. Пользовательские интерфейсы

Следует задать:

1. Логические характеристики каждого интерфейса между программным продуктом и его пользователями. Сюда входят конфигурационные характеристики (например, требуемые экранные формы, страницы или раскладки окон, содержимое всех отчетов или меню, а также доступность программируемых функциональных клавиш), необходимые для выполнения программных требований.
2. Все аспекты оптимизации интерфейса с персоналом, который должен использовать систему. Они могут просто включать список, как система должна выглядеть с точки зрения пользователя, а как не должна. Например, может быть требование наличия опции полных или кратких сообщений об ошибках. Подобно всем остальным, эти

требования должны быть верифицируемыми, например: «оператор 4-го разряда должен научиться выполнять действие X за Z минут после часового обучения», а не «оператор должен выполнять действие X». (Это также можно задать в «Системных атрибутах программного обеспечения» в разделе под названием «Простота использования»).

2.1.3. Аппаратные интерфейсы

Здесь следует задать логические характеристики каждого интерфейса между программным продуктом и аппаратными компонентами системы. Сюда входят конфигурационные характеристики (число портов, набор инструкций и т.д.). Также освещаются такие моменты, как: какие устройства поддерживаются, как они поддерживаются, а также протоколы. Например, поддержка терминала может задавать поддержку полноэкранного интерфейса, в противоположность построчному вводу.

2.1.4. Программные интерфейсы

Следует задать использование прочих необходимых программных продуктов (например, система управления данными, операционная система или математический пакет), а также интерфейсы с другими прикладными системами (например, связь между системой приема оплаты счетов и общей бухгалтерской системой). Для каждого необходимого программного продукта следует предоставить следующую информацию:

- Название;
- Мнемоника;
- Номер спецификации;
- Номер версии;
- Источник.

Для каждого интерфейса следует предоставить следующую информацию:

- Обсуждение назначения интерфейсного программного обеспечения с точки зрения программного продукта.
- Определение интерфейса в терминах содержания и формата сообщений. Не обязательно детально описывать хорошо документированные интерфейсы, но требуется ссылка на документ, определяющий требуемый интерфейс.

2.1.5. Коммуникационные интерфейсы

Следует задать различные интерфейсы коммуникаций: протоколы локальных сетей

2.1.6. Ограничения по памяти

Следует задать все значимые характеристики и ограничения, касающиеся первичной и вторичной памяти.

2.1.7. Операции

Следует задать обычные и специфические операции, которые требуются пользователю, например:

1. Различные модели операций в организации пользователя (например, операции, инициируемые пользователем);
2. Периоды интерактивных операций и периоды операций, не требующих ручного вмешательства;
3. Функции поддержки обработки данных;
4. Операции резервного копирования и восстановления.

Примечание. Иногда они задаются как часть раздела «Пользовательские Интерфейсы».

2.1.8. Требования к адаптации на мест

Следует:

1. Определить требования ко всем данным или последовательностям инициализации, специфичным для данного места, задачи или режима работы (например, таблицы значений, безопасные пределы и т.д.);
2. Задать особенности, относящиеся к месту или задаче, которые следует модифицировать с целью адаптации программного обеспечения к конкретной инсталляции.

Функции продукта (2.2 SRS)

В этом подразделе SRS следует представить сводку основных функций, выполняемых системой. Например, SRS для бухгалтерской программы может посвятить эту часть работе со счетами, обслуживанию клиентов и подготовке платежных поручений, не вдаваясь в обширную детализацию этих функций.

Иногда сводка функций, необходимых для данной части, берется прямо из соответствующего раздела спецификации верхнего уровня (если она есть), которая размещает некоторые функции в программном продукте. Заметим для ясности, что:

1. Функции должны быть организованы таким образом, чтобы сделать перечень функций понятным потребителю или другим читателям при первом прочтении документа.
2. Можно использовать текстовые или графические методики для представления различных функций и отношений между ними. Подобные диаграммы не должны представлять реализацию продукта, а лишь показывать логические взаимосвязи между переменными.

Пользовательские характеристики (2.3 SRS)

Этот подраздел SRS должен описывать общие характеристики предполагаемых пользователей, включая уровень образования, опыт и техническую грамотность. В нем не следует устанавливать специфические требования, но следует привести причины, по которым некоторые специфические требования заданы далее в Разделе 3 SRS.

Ограничения (2.4 SRS)

В этом подразделе должны быть приведены общие описания всего того, что может ограничить действия разработчика. Они включают:

1. Правовые вопросы;
2. Аппаратные ограничения (например, требования к длительности сигналов);
3. Интерфейсы с другими приложениями;
4. Параллельные операции;
5. Функции аудита;
6. Функции управления;
7. Языковые ограничения высшего порядка;
8. Протоколы синхронизации сигналов (например, XON-XOFF, ACK-NACK);
9. Требования к надежности;
10. Критичность приложения;
11. Соображения безопасности и секретности.

Предположения и зависимости (2.5 SRS)

В этом подразделе следует перечислить все факторы, которые влияют на требования, устанавливаемые SRS. Эти факторы не являются проектными ограничениями, а скорее относятся к их изменениям, которые могут повлиять на требования SRS. Например, может быть сделано предположение, что некоторая операционная система будет доступна для оборудования, на которое ориентирован программный продукт. Если в действительности операционная система недоступна, потребуется соответствующее изменение SRS.

Распределение требований (2.6 SRS)

Этот подраздел должен представлять требования, которые могут быть отложены до будущих версий системы.

СПЕЦИФИЧЕСКИЕ ТРЕБОВАНИЯ (РАЗДЕЛ 3 SRS)

Этот раздел SRS должен содержать все требования к программному обеспечению на уровне детализации, достаточном, чтобы позволить разработчикам создать систему, удовлетворяющую этим ограничениям, а тестерам – проверить, удовлетворяет ли система этим ограничениям. В данном разделе каждое требование должно быть ориентировано на пользователей, операторов или другие системы, внешние по отношению к данной. Эти требования должны включать минимальное описание для каждого ввода в систему, каждого ответа системы, а также всех функций, выполняемых системой в ответ на ввод или для поддержки вывода. Поскольку зачастую эта часть SRS является самой большой и наиболее важной, применяются следующие принципы:

1. Специфические требования должны соответствовать характеристикам, описанным в 4.3.
2. Специфические требования должны снабжаться перекрестными ссылками на ранние документы.
3. Все требования должны быть уникально идентифицируемыми.
4. Следует уделить особое внимание организации требований с целью достижения максимальной читабельности.

Перед изучением способов организации требований полезно рассмотреть различные элементы, включающие в себя требования, которые описаны в подразделах с 3.1 по 3.7.

3.1.Внешние интерфейсы

Здесь должно быть детальное описание всех входных и выходных данных программной системы. Оно должно дополнять описания интерфейсов из 5.2 и не содержать повторяющейся информации. Оно должно иметь формат и содержание, как указано ниже:

1. Имя элемента;
2. Описание назначения;
3. Источник входных данных или получатель выходных;
4. допустимый диапазон, точность и отклонения;
5. Единицы измерения;
6. Временные диаграммы;
7. Взаимосвязи с другими входами/выходами;
8. Форматы и организация экранов;
9. Форматы и организация окон;

10. Форматы данных;
11. Форматы команд;
12. Завершающие сообщения.

3.2. Функции

Функциональные требования должны определять фундаментальные действия, которые должны выполняться программным обеспечением для приема и обработки входных данных, а также обработки и вывода выходных данных. Они обычно перечисляются в виде предложений, начинающихся со слов «Система должна…».

Они включают:

1. Проверку допустимости входных значений;
2. Точный порядок действий;
3. Реакцию на нештатные ситуации, включающие:
    1. Переполнение;
    2. Коммуникационные проблемы;
    3. Обработку ошибок и восстановление;
4. Влияние параметров;
5. Взаимосвязь между входными и выходными данными, включая:
    1. Порядок ввода/вывода;
    2. Формулы преобразования входных данных в выходные

Может оказаться удобным разделение функциональных требований на подфункции или подпроцессы. Это не значит, что программный проект будет разделен таким же образом.

3.3. Требования к производительности

Этот подраздел должен задавать как статические, так и динамические численные требования, предъявляемые в целом к программному обеспечению или к взаимодействию человека с программой. Статические численные требования могут включать следующие:

1. Число поддерживаемых терминалов;
2. Число одновременно поддерживаемых пользователей;
3. Объем и тип обрабатываемой информации.

Статические численные требования иногда оформляются в виде отдельного раздела.

Динамические численные требования могут включать, например, число транзакций и задач, или объем данных, обрабатываемых в некоторый период данных в условиях как нормальной, так и пиковой нагрузки.

Все эти требования следует формулировать в терминах измеримых величин.

Например:

95% транзакций должны обрабатываться менее чем за 1 секунду

вместо:

Оператор не должен ждать, пока завершится транзакция.

Примечание. Численные границы, применимые к отдельной функции, обычно задаются в описании данной функции, в подразделе обработки.

3.4. Логические требования к базе данных

Здесь следует задать логические требования к информации, которая должна размещаться в базе данных. Они могут включать следующие:

- Типы информации, используемой различными функциями;
- Частоту использования;
- Возможность доступа;
- Сущности и отношения между ними;
- Ограничения целостности;
- Требования к хранению данных.

3.5. Ограничения проектирования

Здесь задаются ограничения проектирования, налагаемые другими стандартами, аппаратурой и т.д.

3.5.1. Соответствие стандартам

Этот подраздел должен задавать ограничения, вытекающие из существующих стандартов и правил. Они могут включать следующее:

1. Форматы отчетов;
2. Именование данных;
3. Бухгалтерские процедуры;
4. Протоколирование работы.

Например, требования к программному обеспечению по трассировке вычислительных действий. Подобные трассировки необходимы для некоторых приложений для выполнения

требований правовых или финансовых стандартов. Требование протоколирования работы может, например, устанавливать, что при изменениях в платежной ведомости прежнее и новое значения должны записываться в файл трассировки.

3.6. Атрибуты программной системы

Некоторые атрибуты программного обеспечения могут служить требованиями. Важно, чтобы требуемые атрибуты были заданы таким образом, чтобы можно было объективно проверить выполнение данного требования. В подразделах с 3.6.1 по 3.6.5 приведен список примеров.

3.6.1. Надежность

Следует перечислить факторы, необходимые для установления требуемого уровня надежности программного обеспечения.

3.6.2. Доступность

Следует перечислить факторы, призванные гарантировать определенный уровень доступности системы, такие как контрольные точки, восстановление и перезапуск.

3.6.3. Безопасность

Следует перечислить факторы, защищающие программное обеспечение от случайного или злонамеренного доступа, использования, модификации, разрушения или разглашения. Специфические требования в этой области включают необходимость:

1. Использования криптографии;
2. Хранение логов или истории;
3. Назначать некоторые функции различным модулям;
4. Ограничивать коммуникации между некоторыми областями программы;
5. Проверять целостность данных для критических переменных.

3.6.4. Поддерживаемость

Следует перечислить атрибуты программного обеспечения, относящиеся к легкости поддержки самого программного обеспечения. Это могут быть некоторые требования, относящиеся к модульности, интерфейсам, сложности и т.д. Не следует помещать здесь требования лишь потому, что принято считать их хорошим тоном разработки.

3.6.5. Переносимость

Следует перечислить атрибуты программного обеспечения, касающиеся легкости переноса программного обеспечения на другие компьютеры и/или операционные системы. Они могут включать следующее:

1. Доля компонентов с машинно-зависимым кодом;
2. Доля машинно-зависимого кода;
3. Использование испытанного переносимого языка;
4. Использование особенного компилятора или подмножества языка;
5. Использование особенной операционной системы.

3.7. Организация специфических требований

Для любой нетривиальной системы детальные требования обширно разрастаются. Поэтому рекомендуется тщательный подход к их организации в виде, оптимальном для понимания. Не существует единой организации, оптимальной для всех систем. Различные классы систем придерживаются различной организации требований в Разделе 3 SRS. Некоторые способы организации описаны в подразделах с 3.7.1 по 3.7.7.

3.7.1. Режим системы

Поведение некоторых систем кардинально различается в зависимости от режима системы. Например, система управления может иметь различные наборы функций для разных режимов: обучение, нормальный или аварийный. При организации этого раздела по режимам, можно использовать планы A.1 или A.2. Выбор определяется тем, зависят ли от режима интерфейсы и производительность.

3.7.2. Класс пользователя

Некоторые системы обеспечивают различные наборы функций для разных классов пользователей. Например, система управления лифтом предоставляет различные возможности пассажирам, обслуживающему персоналу и пожарным. При организации этого раздела по классам пользователей следует использовать план A.3.

3.7.3. Объекты

Объекты – это сущности реального мира, которые имеют соответствие в системе. Например, в системе мониторинга пациентов в число объектов входят пациенты, датчики, медсестры, кабинеты, врачи, медикаменты и т.д. С каждым объектом связаны наборы атрибутов (данного объекта) и функций (выполняемых данным объектом). Эти функции называются также сервисами, методами или процессами. При организации этого раздела по объектам следует использовать план

А.4. Заметим, что наборы объектов могут иметь общие атрибуты и сервисы. Они группируются вместе как классы.

3.7.4. Функциональные возможности

Функциональная возможность – это внешний сервис системы, который может потребовать последовательность ввода данных для достижения желаемого результата. Например, в телефонной системе функциональные возможности включают местный вызов, переадресацию звонка и конференцию. Обычно функциональные возможности описывают в виде последовательности пар запрос-ответ. При организации этого раздела в соответствии с функциональными возможностями следует использовать план А.5.

3.7.5. Внешние воздействия

Некоторые системы могут быть лучше организованы путем описания их функций в терминах внешних воздействий. Например, функции системы автоматического приземления самолета могут быть разбиты на секции для потери тяги, порывов ветра, неожиданной смены курса, избыточной вертикальной скорости и т.д. При организации этого раздела в соответствии с внешними воздействиями следует использовать план А.6.

3.7.6. Отклики

Некоторые системы могут быть лучше организованы путем описания всех функций, поддерживающих генерацию откликов. Например, функции системы управления персоналом могут быть разбиты на секции, соответствующие всем функциям, связанным с генерацией чеков оплаты, всем функциям, связанным с генерацией текущего списка сотрудников, и т.д. Следует использовать план А.6, в котором все внешние воздействия заменены откликами.

3.7.7. Функциональная иерархия

Если ни одна из вышеперечисленных организационных схем не подходит, общая функциональность может быть организована в виде иерархии функций, организованных в соответствии с входными данными, выходными данными или внутренними данными. Могут быть использованы диаграммы потоков данных и словари данных, чтобы показать взаимосвязи между функциями и данными. При организации этого раздела в соответствии с функциональной иерархией следует использовать план А.7.

3.8. Дополнительные комментарии

В процессе работы над SRS могут оказаться применимыми более одной организационной схемы из перечисленных в 3.7.7. В таких случаях организуйте специфические требования по нескольким иерархиям, приспособленным под специфические нужды системы, для которой разрабатываются спецификации. Например, см. A.8 как пример организации, комбинирующей классы пользователей и функциональные возможности. Все дополнительные требования могут быть размещены в отдельном разделе в конце SRS.

Имеется множество нотаций, методик и автоматизированных инструментов для поддержки документирования требований. В основном они полезны в части организационной функции. Например, при организации по режимам могут оказаться полезными конечные автоматы диаграммы состояний; при организации по объектам может оказаться полезным объектно-ориентированный анализ; при организации по функциональным возможностям могут оказаться полезными последовательности запрос-ответ, а при организации по функциональной иерархии могут пригодиться диаграммы потоков данных и словари данных.

В любом из планов, описанных с A.1 по A.8, разделы с именем «Функциональное требование i» могут быть описаны на естественном языке, на псевдокоде, на языке описания систем либо в виде четырех подразделов с именами «Введение», «Входные данные», «Обработка» и «Выходные данные».

4. ВСПОМОГАТЕЛЬНАЯ ИНФОРМАЦИЯ

Вспомогательная информация делает SRS более удобочитаемой. Она включает следующее:

- Оглавление;
- Индекс;
- Приложения.

4.1. Оглавление и индекс

Оглавление и индекс крайне важны и должны следовать общим принципам композиции.

4.2. Приложения

Приложения не всегда рассматриваются как часть SRS и не всегда необходимы. Они могут включать:

- Примеры форматов входных и выходных данных, описания примеров ценового анализа или результаты пользовательских обзоров;

- Вспомогательная и дополнительная информация, которая может помочь читателю SRS;
- Описание проблемы, решаемой программным обеспечением;
- Специальные инструкции по упаковке для кода и носителей для соответствия требованиям по безопасности, экспорту начальной загрузке и т.д.

Если приложения включены в состав SRS, следует явно указать, являются ли они частью требований.

Контрольные вопросы:

23. Опишите из чего состоит общий раздел документации?
24. Опешите специфический раздел требований
25. Опешите раздел приложений
26. Опишите раздел индекса

Список использованных источников:

5. Государственный стандарт Российской Федерации. Информационная технология. Сопровождение программных средств
6. Основы маркетинга учебное пособие Суркова Е.В.
7. Рекомендуемая стандартом IEEE 830структура SRS

ЛЕКЦИЯ 12

Тема: IEEE Standard SRS Template (спецификации требования к программному обеспечению). Раздел специфические требования к программному продукту. Приложения . Индексы.

Цель: изучить спецификацию требований к ПО.

Содержание SRS:

11. Введение
    1. Назначение
    2. Область применения
    3. Определения, акронимы и сокращения
    4. Ссылки
    5. Обзор
12. Общее описание
    1. Позиционирование продукта
    2. Функции продукта
    3. Пользовательские характеристики
    4. Ограничения
    5. Предположения и зависимости
13. Специфические требования
14. Приложения

15. Индекс

СПЕЦИФИЧЕСКИЕ ТРЕБОВАНИЯ (РАЗДЕЛ 3 SRS)

Этот раздел SRS должен содержать все требования к программному обеспечению на уровне детализации, достаточном, чтобы позволить разработчикам создать систему, удовлетворяющую этим ограничениям, а тестерам – проверить, удовлетворяет ли система этим ограничениям. В данном разделе каждое требование должно быть ориентировано на пользователей, операторов или другие системы, внешние по отношению к данной. Эти требования должны включать минимальное описание для каждого ввода в систему, каждого ответа системы, а также всех функций, выполняемых системой в ответ на ввод или для поддержки вывода. Поскольку зачастую эта часть SRS является самой большой и наиболее важной, применяются следующие принципы:

5. Специфические требования должны соответствовать характеристикам, описанным в 4.3.
6. Специфические требования должны снабжаться перекрестными ссылками на ранние документы.
7. Все требования должны быть уникально идентифицируемыми.
8. Следует уделить особое внимание организации требований с целью достижения максимальной читабельности.

Перед изучением способов организации требований полезно рассмотреть различные элементы, включающие в себя требования, которые описаны в подразделах с 3.1 по 3.7.

3.1.Внешние интерфейсы

Здесь должно быть детальное описание всех входных и выходных данных программной системы. Оно должно дополнять описания интерфейсов из 5.2 и не содержать повторяющейся информации. Оно должно иметь формат и содержание, как указано ниже:

13. Имя элемента;
14. Описание назначения;
15. Источник входных данных или получатель выходных;
16. допустимый диапазон, точность и отклонения;
17. Единицы измерения;
18. Временные диаграммы;
19. Взаимосвязи с другими входами/выходами;

20. Форматы и организация экранов;
21. Форматы и организация окон;
22. Форматы данных;
23. Форматы команд;
24. Завершающие сообщения.

3.2. Функции

Функциональные требования должны определять фундаментальные действия, которые должны выполняться программным обеспечением для приема и обработки входных данных, а также обработки и вывода выходных данных. Они обычно перечисляются в виде предложений, начинающихся со слов «Система должна…».

Они включают:

6. Проверку допустимости входных значений;
7. Точный порядок действий;
8. Реакцию на нештатные ситуации, включающие:
    1. Переполнение;
    2. Коммуникационные проблемы;
    3. Обработку ошибок и восстановление;
9. Влияние параметров;
10. Взаимосвязь между входными и выходными данными, включая:
    1. Порядок ввода/вывода;
    2. Формулы преобразования входных данных в выходные

Может оказаться удобным разделение функциональных требований на подфункции или подпроцессы. Это не значит, что программный проект будет разделен таким же образом.

3.3. Требования к производительности

Этот подраздел должен задавать как статические, так и динамические численные требования, предъявляемые в целом к программному обеспечению или к взаимодействию человека с программой. Статические численные требования могут включать следующие:

4. Число поддерживаемых терминалов;
5. Число одновременно поддерживаемых пользователей;
6. Объем и тип обрабатываемой информации.

Статические численные требования иногда оформляются в виде отдельного раздела.

Динамические численные требования могут включать, например, число транзакций и задач, или объем данных, обрабатываемых в некоторый период данных в условиях как нормальной, так и пиковой нагрузки.

Все эти требования следует формулировать в терминах измеримых величин.

Например:

95% транзакций должны обрабатываться менее чем за 1 секунду

вместо:

Оператор не должен ждать, пока завершится транзакция.

Примечание. Численные границы, применимые к отдельной функции, обычно задаются в описании данной функции, в подразделе обработки.

3.4. Логические требования к базе данных

Здесь следует задать логические требования к информации, которая должна размещаться в базе данных. Они могут включать следующие:

- Типы информации, используемой различными функциями;
- Частоту использования;
- Возможность доступа;
- Сущности и отношения между ними;
- Ограничения целостности;
- Требования к хранению данных.

3.5. Ограничения проектирования

Здесь задаются ограничения проектирования, налагаемые другими стандартами, аппаратурой и т.д.

3.5.1. Соответствие стандартам

Этот подраздел должен задавать ограничения, вытекающие из существующих стандартов и правил. Они могут включать следующее:

5. Форматы отчетов;
6. Именование данных;
7. Бухгалтерские процедуры;
8. Протоколирование работы.

Например, требования к программному обеспечению по трассировке вычислительных действий. Подобные трассировки необходимы для некоторых приложений для выполнения требований правовых или финансовых стандартов. Требование протоколирования работы

может, например, устанавливать, что при изменениях в платежной ведомости прежнее и новое значения должны записываться в файл трассировки.

3.6. Атрибуты программной системы

Некоторые атрибуты программного обеспечения могут служить требованиями. Важно, чтобы требуемые атрибуты были заданы таким образом, чтобы можно было объективно проверить выполнение данного требования. В подразделах с 3.6.1 по 3.6.5 приведен список примеров.

3.6.1. Надежность

Следует перечислить факторы, необходимые для установления требуемого уровня надежности программного обеспечения.

3.6.2. Доступность

Следует перечислить факторы, призванные гарантировать определенный уровень доступности системы, такие как контрольные точки, восстановление и перезапуск.

3.6.3. Безопасность

Следует перечислить факторы, защищающие программное обеспечение от случайного или злонамеренного доступа, использования, модификации, разрушения или разглашения. Специфические требования в этой области включают необходимость:

6. Использования криптографии;
7. Хранение логов или истории;
8. Назначать некоторые функции различным модулям;
9. Ограничивать коммуникации между некоторыми областями программы;
10. Проверять целостность данных для критических переменных.

3.6.4. Поддерживаемость

Следует перечислить атрибуты программного обеспечения, относящиеся к легкости поддержки самого программного обеспечения. Это могут быть некоторые требования, относящиеся к модульности, интерфейсам, сложности и т.д. Не следует помещать здесь требования лишь потому, что принято считать их хорошим тоном разработки.

3.6.5. Переносимость

Следует перечислить атрибуты программного обеспечения, касающиеся легкости переноса программного обеспечения на другие компьютеры и/или операционные системы. Они могут включать следующее:

6. Доля компонентов с машинно-зависимым кодом;
7. Доля машинно-зависимого кода;
8. Использование испытанного переносимого языка;
9. Использование особенного компилятора или подмножества языка;
10. Использование особенной операционной системы.

3.7. Организация специфических требований

Для любой нетривиальной системы детальные требования обширно разрастаются. Поэтому рекомендуется тщательный подход к их организации в виде, оптимальном для понимания. Не существует единой организации, оптимальной для всех систем. Различные классы систем придерживаются различной организации требований в Разделе 3 SRS. Некоторые способы организации описаны в подразделах с 3.7.1 по 3.7.7.

3.7.1. Режим системы

Поведение некоторых систем кардинально различается в зависимости от режима системы. Например, система управления может иметь различные наборы функций для разных режимов: обучение, нормальный или аварийный. При организации этого раздела по режимам, можно использовать планы A.1 или A.2. Выбор определяется тем, зависят ли от режима интерфейсы и производительность.

3.7.2. Класс пользователя

Некоторые системы обеспечивают различные наборы функций для разных классов пользователей. Например, система управления лифтом предоставляет различные возможности пассажирам, обслуживающему персоналу и пожарным. При организации этого раздела по классам пользователей следует использовать план A.3.

3.7.3. Объекты

Объекты – это сущности реального мира, которые имеют соответствие в системе. Например, в системе мониторинга пациентов в число объектов входят пациенты, датчики, медсестры, кабинеты, врачи, медикаменты и т.д. С каждым объектом связаны наборы атрибутов (данного объекта) и функций (выполняемых данным объектом). Эти функции называются также сервисами, методами или процессами. При организации этого раздела по объектам следует использовать план

A.4. Заметим, что наборы объектов могут иметь общие атрибуты и сервисы. Они группируются вместе как классы.

3.7.4. Функциональные возможности

Функциональная возможность – это внешний сервис системы, который может потребовать последовательность ввода данных для достижения желаемого результата. Например, в телефонной системе функциональные возможности включают местный вызов, переадресацию звонка и конференцию. Обычно функциональные возможности описывают в виде последовательности пар запрос-ответ. При организации этого раздела в соответствии с функциональными возможностями следует использовать план A.5.

3.7.5. Внешние воздействия

Некоторые системы могут быть лучше организованы путем описания их функций в терминах внешних воздействий. Например, функции системы автоматического приземления самолета могут быть разбиты на секции для потери тяги, порывов ветра, неожиданной смены курса, избыточной вертикальной скорости и т.д. При организации этого раздела в соответствии с внешними воздействиями следует использовать план A.6.

3.7.6. Отклики

Некоторые системы могут быть лучше организованы путем описания всех функций, поддерживающих генерацию откликов. Например, функции системы управления персоналом могут быть разбиты на секции, соответствующие всем функциям, связанным с генерацией чеков оплаты, всем функциям, связанным с генерацией текущего списка сотрудников, и т.д. Следует использовать план A.6, в котором все внешние воздействия заменены откликами.

3.7.7. Функциональная иерархия

Если ни одна из вышеперечисленных организационных схем не подходит, общая функциональность может быть организована в виде иерархии функций, организованных в соответствии с входными данными, выходными данными или внутренними данными. Могут быть использованы диаграммы потоков данных и словари данных, чтобы показать взаимосвязи между функциями и данными. При организации этого раздела в соответствии с функциональной иерархией следует использовать план A.7.

3.8. Дополнительные комментарии

В процессе работы над SRS могут оказаться применимыми более одной организационной схемы из перечисленных в 3.7.7. В таких случаях организуйте специфические требования по нескольким иерархиям, приспособленным под специфические нужды системы, для которой разрабатываются спецификации. Например, см. A.8 как пример организации, комбинирующей классы пользователей и функциональные возможности. Все дополнительные требования могут быть размещены в отдельном разделе в конце SRS.

Имеется множество нотаций, методик и автоматизированных инструментов для поддержки документирования требований. В основном они полезны в части организационной функции. Например, при организации по режимам могут оказаться полезными конечные автоматы диаграммы состояний; при организации по объектам может оказаться полезным объектно-ориентированный анализ; при организации по функциональным возможностям могут оказаться полезными последовательности запрос-ответ, а при организации по функциональной иерархии могут пригодиться диаграммы потоков данных и словари данных.

В любом из планов, описанных с A.1 по A.8, разделы с именем «Функциональное требование i» могут быть описаны на естественном языке, на псевдокоде, на языке описания систем либо в виде четырех подразделов с именами «Введение», «Входные данные», «Обработка» и «Выходные данные».

4. ВСПОМОГАТЕЛЬНАЯ ИНФОРМАЦИЯ

Вспомогательная информация делает SRS более удобочитаемой. Она включает следующее:

- Оглавление;
- Индекс;
- Приложения.

4.1. Оглавление и индекс

Оглавление и индекс крайне важны и должны следовать общим принципам композиции.

4.2. Приложения

Приложения не всегда рассматриваются как часть SRS и не всегда необходимы. Они могут включать:

- Примеры форматов входных и выходных данных, описания примеров ценового анализа или результаты пользовательских обзоров;

- Вспомогательная и дополнительная информация, которая может помочь читателю SRS;
- Описание проблемы, решаемой программным обеспечением;
- Специальные инструкции по упаковке для кода и носителей для соответствия требованиям по безопасности, экспорту начальной загрузке и т.д.

Если приложения включены в состав SRS, следует явно указать, являются ли они частью требований.

Контрольные вопросы:

27. Опишите из чего состоит общий раздел документации?
28. Опешите специфический раздел требований
29. Опешите раздел приложений
30. Опишите раздел индекса

Список использованных источников:

8. Государственный стандарт Российской Федерации. Информационная технология. Сопровождение программных средств
9. Основы маркетинга учебное пособие Суркова Е.В.
10. Рекомендуемая стандартом IEEE 830структура SRS

ЛЕКЦИЯ 13

Тема: организация испытаний и оценка испытания ПП. Категории испытаний.

Цель: изучить спецификацию требований к ПО.

Испытания продукции - один из наиболее важных и интересных этапов постановки продукции на производство.

Характеристики свойств объекта при испытаниях *могут оцениваться*, если целью испытаний является получение количественных или качественных характеристик, а *могут контролироваться*, если целью испытаний является установление соответствия характеристик объекта заданным требованиям.

В этом случае испытания сводятся к контролю. Поэтому ряд видов испытаний являются контрольными, в процессе которых выполняется процесс контроля качества продукции.

Цель испытания – определение степени соответствия созданного комплекса программ

Виды испытаний:

1. ***Испытание опытного образца ПС*** на полное соответствие требованиям ТЗ.
2. **Испытание рабочей версии ПС**, адаптируемое к конкретным условиям применения.
3. **Испытание** версии (**модернизированного ПС** при сопровождении)

Категории испытаний

Назначение испытаний	**Исследовательские**	Испытания, проводимые *для изучения определенных характеристик свойств объекта*
	Контрольные	Испытания, проводимые *для контроля качества объекта*
	Сравнительные	Испытания аналогичных по характеристикам или одинаковых объектов, проводимые в идентичных условиях *для сравнения характеристик их свойств*
	Определительные	Испытания, проводимые *для определения значений характеристик объекта с заданными значениями показателей точности* и (или) достоверности
Уровень проведения испытаний	**Государственные**	Испытания установленных важнейших видов продукции, проводимые головной организацией по государственным испытаниям, или приемочные испытания, проводимые *государственной комиссией или испытательной организацией*, которой предоставлено право их проведения
	Межведомственные	Испытания продукции, *проводимые комиссией из представителей нескольких заинтересованных министерств для приемки составных частей объекта, разрабатываемого совместно несколькими ведомствами*
	Ведомственные	Испытания, *проводимые комиссией из представителей заинтересованного министерства или ведомства*
Этапы разработки продукции	**Доводочные**	Исследовательские испытания, *проводимые* при разработке *продукции с целью оценки влияния вносимых в нее изменений для достижения заданных значений показателей ее качества*
	Предварительные	*Контрольные испытания опытных образцов* и (или) опытных партий продукции *с целью определения возможности их предъявления на приемочные испытания*
	Приемочные	*Контрольные испытания опытных образцов*, опытных партий продукции или изделии единичного производства, *проводимые соответственно с целью решения вопроса о целесообразности постановки этой продукции на производство*
Испытания готовой продукции	**Квалификационные**	*Контрольные испытания первой промышленной партии, проводимые с целью оценки готовности предприятия к выпуску продукции* данного типа в заданном объеме
	Предъявительские	Контрольные испытания продукции, проводимые службой технического контроля предприятия-изготовителя перед предъявлением ее для приемки заказчика, потребителя пли других

		органов приемки
	Приемо-сдаточные	Контрольные испытания продукции при приемочном контроле
	Периодические	Контрольные испытания выпускаемой продукции, проводимые в объемах и в сроки, установленные нормативно-технической документацией, с целью контроля стабильности качества продукции и возможности продолжения ее выпуска
	Инспекционные	Контрольные испытания установленных видов выпускаемой продукции, проводимые в выборочном порядке с целью контроля стабильности качества продукции специально уполномоченными организациями
	Типовые	Контрольные испытания выпускаемой продукции, проводимые с целью оценки эффективности и целесообразности вносимых изменений в конструкцию, рецептуру или технологический процесс
	Аттестационные	Испытания, проводимые для оценки уровня качества продукции при ее аттестации по категориям качества
	Сертификационные	Контрольные испытания продукции, проводимые с целью установления соответствия характеристик свойств национальным и (или) международным нормативно-техническим документам
Условия и место проведения испытаний	**Лабораторные**	Испытания объекта, проводимые в лабораторных условиях
	Стендовые	Испытания объекта, проводимые на испытательном оборудовании
	Полигонные	Испытания объекта, проводимые на испытательном полигоне
	Натурные	Испытания объекта в условиях, соответствующих условиям его использования по прямому назначению с непосредственным оцениванием или контролем определяемых характеристик свойств объекта
	Испытания с использованием моделей	Испытания с использованием моделей включают проведение расчетов на математических или физико-математических моделях объекта испытаний и (или) воздействий на него в сочетании с натурными испытаниями объекта и его составных частей (опытно-теоретический метод испытаний), а также применение физической модели объекта испытаний или его составных частей
	Эксплуатационные	Испытания объекта, проводимые при эксплуатации
Продолжительность испытаний	**Нормальные**	Испытания, методы и условия проведения которых обеспечивают получение необходимого объема информации о характеристиках свойств объекта в такой же интервал времени, как и в предусмотренных условиях эксплуатации.
	Ускоренные	Испытания, методы и условия проведения которых обеспечивают получение необходимой информации о характеристиках свойств объекта в более короткий срок, чем при нормальных испытаниях
	Сокращенные	Испытания, проводимые по сокращенной программе
Вид воздействия	**Механические**	Испытания на воздействие механических факторов
	Климатические	Испытания на воздействие климатических факторов

	Термические	Испытания на воздействие термических факторов
	Радиационные	Испытания на воздействие радиационных факторов
	Электрические	Испытания на воздействие электрического напряжения, тока или ноля
	Электромагнитные	Испытания на воздействие электромагнитных полей
	Магнитные	Испытания на воздействие магнитного поля
	Химические	Испытания на воздействие специальных сред
	Биологические	Испытания на воздействие биологических факторов
Результат воздействия	**Неразрушающие**	Испытания с применением неразрушающих методов контроля
	Разрушающие	Испытания с применением разрушающие методов контроля
	Испытания на прочность	Испытания, проводимые для определения значений воздействующих факторов, вызывающих выход значений характеристик свойств объекта за установленные пределы или его разрушение
	Испытания на устойчивость	Испытания, проводимые для контроля способности изделия выполнять свои функции и сохранять значения параметров в пределах установленных норм во время действия на него определенных факторов
Определяемые характеристики объекта	**Функциональные**	Испытания, проводимые с целью определения значений показателей назначения объекта
	Испытания на надежность	Испытания, проводимые для определения показателей надежности в заданных условиях
	Испытания на безопасность	Испытания, проводимые для определения показателей безопасности продукции
	Испытания на транспортабельность	Испытания, проводимые для определения сохраняемости изделия после транспортировки
	Граничные испытания	Испытания, проводимые для определения зависимостей между предельно допустимыми значениями параметров объекта и режимом эксплуатации
	Технологические испытания	Испытания, проводимые при изготовлении продукции с целью оценки ее технологичности

Контрольные вопросы:

31. Назовите виды воздействий?

32. Назовите какие бывают результаты воздействий?

33. Назовите какие бывают продолжительности испытаний?

34. Назовите какие бывают условия и места испытаний?

35. Назовите какие бывают этапы разработки?

36. Назовите какие бывают уровни проведения испытаний?

Список использованных источников:

11. Государственный стандарт Российской Федерации. Информационная технология. Сопровождение программных средств

12. Основы маркетинга учебное пособие Суркова Е.В.

13. Рекомендуемая стандартом IEEE 830структура SRS

ЛЕКЦИЯ 14

Тема: Сопровождение ПС. Конфигурационное управление. Сертификация. Стандарты сертификации

Цель: изучить принцип сертификации и конфигурационного управления ПС

СОПРОВОЖДЕНИЕ

Сопровождение ПС -Процесс модификации ПС, обусловленный необходимостью устранения выявленных в нем ошибок, и/или изменение функциональных возможностей.

Это то когда наш продукт уже какое-то время используется заказчикам и его нужно поддерживать в работоспособном состоянии оговоренный срок с заказчиком

Что бы осуществлять сопровождение не только после выпуска ПС, но и при его разработки, находить баги и устранять их вовремя, вносить изменения и обновления в версии разработки для этого используют КОНФИГУРАЦИОННОЕ УПРАВЛЕНИЕ

Конфигурационное управление Процесс применения административных и технических процедур на всем протяжении жизненного цикла для:

1. *идентификации*, определения и базирования единиц ПС в *информационных системах*;
2. *управления модификацией* и выпуском версий ПС;
3. *фиксирования и сообщения о состоянии версии* ПС;
4. *для управления и контролирования хранения обращения и поставок* ПС.

Все то, что может покажет состояние на данный момент нашего ПО

Цель конфигурации : – обеспечить управляемое и контролируемое развитие структуры ПС.

Обеспечить управляемую и вирируемую структуру ПС , чтобы мы в любой момент могли откатить версию ПО на более ранний срок, если произошла глобальная ошибка.

СЕРТИФИКАЦИЯ ПС

Цели сертификации ПС:

1) ***Основная*** – защита интересов пользователей:
   - Контроль качества.
   - Обеспечение высоких потребительских свойств.
   - Повышение эффективности затрат.
2) ***Формальная*** – выдача сертификата:
   - Полнота, точность эталонных данных.
   - Адекватные показатели качества ПС.
   - Методологии интерпретации данных.

Сертификат соответствия – *документ*, издаваемый в соответствии с правилами системы сертификации, *удостоверяющий*, что обеспечивается необходимая уверенность в том, что должным образом идентифицированная продукция, *процесс или услуга соответствуют конкретным стандартам или другим нормативным документам*.

Задачи сертификации: *распределение экономической и юридической ответственности* между испытателями, разработчиками и заказчиками за качество сертифицированной продукции и за возможный ущерб при ее несоответствии документированному и объявленному качеству.

Т.е. кто будет нести ответственность, включая поломки, за нанесенный ущерб.

Сертификация бывает 2 видов:

1) Обязательная.
2) Добровольная.

Организационная структура системы сертификации

1) *Госстандарт* РФ.
2) *Ведомственные органыте* же функции, но в ограниченном объеме и для конкретных классов продукции.
3) *Испытательные лаборатории* сертификации (ИЛС)

Стандарты сертификации

1) ISO - 0002 : 1983 - общие **термины и определения в области стандартизации** и смежных видах деятельности.
2) ISO - 0025 : 1983 – общие **требования к оценке технической компетентности**.
3) ИЛС ISO - 0038 : 1983 – **общие требования к приемке**
4) ИЛС ISO - 0043 : 1983 - **организация и проведение проверки на компетентность**

5) ISO - 0045 : 1983 - руководящее положение по **представлению результатов испытаний.**

6) ISO - 0049 : 1983 – руководящее положение **по разработке, руководство по качеству для испытаний лаборатории.**

7) ISO - 0054 : 1983 – общие **требования к приемке органов аккредитации**.

8) ISO - 0055 : 1983 – система аккредитации ИЛС**, общие требования к испытательной деятельности**

Контрольные вопросы:

37. Какие цели сертификации вы знаете?
38. Назовите основные стандарты сертификации
39. Виды сертификации?
40. Назовите организационную структуру сертификации?
41. Что такое сопровождение?
42. Цель конфигурации?
43. Что такое конфигурационное управление?
44. Что такое сертификат соответствия?

Список использованных источников:

14. Государственный стандарт Российской Федерации. Информационная технология. Сопровождение программных средств

15. Основы маркетинга учебное пособие Суркова Е.В.

16. Рекомендуемая стандартом IEEE 830структура SRS