UML (Unified Modeling Language) – унифицированный язык моделирования.

Язык UML представляет собой общецелевой язык для моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов.

Лекция № 2

DFD – диаграммы потоков данных. Это методология графического структурного анализа, описывающие внешние источники и адресаты данных, логические функции, потоки данных и хранилища данных.

Принципы разработки графического интерфейса

Основа всего – пользователь.

Программное обеспечение должно разрабатываться с учетом требований и пожеланий пользователя – система должна подстраиваться к пользователю.

Каждый позитивный опыт общения с программой позволяет пользователю расширять область знакомства с программным обеспечением и повышать свой уровень компетентности.

Принципы – это руководство к действию

Три принципы разработки пользовательского интерфейса:

1. Контроль пользователем интерфейса
2. Уменьшение загрузки памяти пользователя
3. Последовательность пользовательского интерфейса

Принципы Хансена:

Принципы таковы:

1. Знать пользователя
2. Сократить запоминание
3. Оптимизировать операции
4. Устранить ошибки

Правило проектирования пользовательского интерфейса

«Делай это проще, но не примитивнее» - Альберт Энтештейн

Правило 1: дать контроль пользователю

Опытные проектировщики позволяют пользователям решать некоторые задачи по собственному усмотрению.

Принципы, которые дают пользователю контроль над системной:

1. Использовать режимы благоразумно
2. Предоставить пользователю возможность выбирать: работать либо мышью, либо клавиатурой, либо их комбинацией
3. Позволить пользователю сфокусировать внимание
4. Демонстрировать сообщения, которые помогут ему в работе
5. Создать условия для немедленных и обратимых действий, а также обратной связи
6. Обеспечить соответствующие пути и выходы
7. Приспосабливайте систему к пользователям с различным уровнем подготовки
8. Сделать пользовательский интерфейс более понятным
9. Дать пользователю возможность настраивать интерфейс по своему вкусу
10. Разрешить пользователю напрямую манипулировать объектами интерфейса

Использовать режимы благоразумно

Режимы – не всегда плохо. Надо позволить человеку самому выбирать нужные ему режимы. Интерфейс должен быть настолько естественным, чтобы пользователю было комфортно работать с ними. Пользователь не думает о переключении в режим вставка или перезаписи при работе в текстовом процессоре – это вполне рационально и естественно.

Возможность работы с клавиатурой использование клавиатуры вместо мыши. Это значит, что пользователю будет легче работать, просто он либо не может ею пользоваться, либо ее у него нет. Панели инструментов созданы, чтобы ускорить работу при использовании мыши. Однако при работе с клавиатурой до них нельзя добраться – для подобных случаев предусмотрены «выпадающие» меню.

Позволить пользователю переключить внимание.

Не вынуждать пользователей заканчивать выполнение начатых последовательностей действий. Дать им выбор – аннулировать или сохранить данные и вернуться назад, где они прервались. Пусть у пользователей останется возможность контролировать процесс работы в программе.

Показывать поясняющие сообщения и тексты

Во всем интерфейсе использовать понятные для пользователя термины. Они не обязаны знать о битах и байтах!

Следует выбрать правильный тон в сообщениях и приглашениях, не менее важно застраховаться от проблем и ошибок. Неудачная терминология и не правильный тон приведут к тому, что пользователи будут винить себя в возникающих ошибках.

Обеспечить немедленные и обратимые действия и обратную связь

Каждый программный продукт должен включить в себя функции UNDO и REDO. Необходимо информировать пользователя о том, что данное действие не может быть отменено, и по возможности разрешить ему альтернативное действие. Постоянно держать человека в курсе происходящего в данный момент.

Предоставлять понятные пути и выходы

Пользователи должны получать удовольствие при работе с интерфейсом любого программного продукта. Даже интерфейсы, применяемые в индустрии, не должны пугать пользователя, он не должен бояться нажиматься нажимать кнопки или переходить на другой экран.

Приспосабливаться к пользователям с разными уровнями навыков

Не «жертвовать» опытными пользователями во благо обычных. Надо предусмотреть для них быстрый доступ к функциям программы. Не утомлять прохождением многочисленных шагов.

Сделать пользовательский интерфейс «прозрачным»

Жать пользователю возможность настроить интерфейс на свой вкус

Прямая связь с ментальной моделью.

Разрешить пользователю прямое манипулирование объектами интерфейса

Объекты должны содержать вес изменений, удалений и прочих работ.

2) Уменьшение загрузки памяти пользоватедя

После регистрации должно быть большое окно, что содержит в себе максимальную информацию с возможностью переходит на дочерние окна с возможность возврата.

Нельзя скрывать элементы.

Не загружать кратковременную память

Автоматический вход после введенной пользователем информации при регистрации.

Полагаться на распознание, а не на повторение

Предусмотреть выбор в виде списка, либо меню, а также предусмотреть списки наиболее важных объектов и документов без введения командных строк

Предоставить визуальные заставки

Визуальная индикация режимов работы.

Предусмотреть установки по умолчанию, команды UNDO и RENDO

Автоматизация действий, связанных с тем, чтобы авторизоваться и регистрироваться а также по установке и разными действиями

Обеспечить ярлыки для интерфейса

Нужно создать быстрый способ открытия программы, чтобы пользователь не прибегал к более долгому открытию

Активировать синтаксис действий с объектами

Должен быть интерфейс, который включает логически и физически распределенный синтаксис для интуитивного понимания пользователем интерфейса.

Использовать метафоры их реального мира

Использовать ассоциации, с которыми пользователь сталкивается в реальном мире, чтобы ему было понятно

Применять раскрытие и объяснение понятий и действий

Замаскировать неиспользуемые функции

Увеличить визуальную ясность.

Яркий и понятный интерфейс, чтобы предоставлять информацию

Правило 3: Сделать интерфейс совместимыми программами

1. Проектирование последовательного интерфейса

Опорные точки при перемещении в интерфейсе (заголовки окно, навигационные карты) и так далее.

1. Общая совместимость всех программ

Похожесть программы с известными аналогами

1. Сохранение результатов взаимодействия

Вводить лишь небольшое число чего-то нового, оставить знакомый интерфейс в других программах

1. Эстетическая привлекательность и цельность

Возможность изменения действия, которое произвел пользователь

1. Интерфейс

Красивый интерфейс приятный глазу

1. Поощрения изучения

Хороший интерфейс, использующий весь функционал

Оценка качество ПО. Стандартизация разработки ПП. Основные программные документы.

Пользователи программного обеспечения испытывают потребности в создании моделей качества программного обеспечения для оценки качества как качественно, так и количественно. Модели качества, которые имеются в настоящее время, в большинстве случаев являются иерархическими моделями на основе критериев качества и связанных с ними показателей (метрик).

Все модели качества могут быть разделены на три категории в соответствии с методами, на основе которых они были созданы

К первому виду можно отнести теоретические модели, основанные на гипотезе отношений между переменными качествами

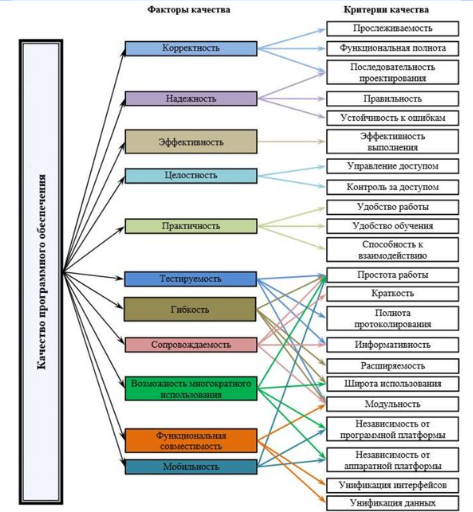
Ко второму виду относятся модели «управления данными», основанные на статистическими анализе

Комбинированная модель, в которой интуиция исследования используется для определения нужного вида модели, а анализ данных используется для определения констант качества.

1. Модель Маккола

Основная модель предназначена для определения полной характеристики качества программного продукта через его различные характеристики:

1. Использование (корректность, надежность, эффективность, целостность, практичность)
2. Модификация (тестируемость, гибкость, сопровождаемость – факторы качества важные для разработки новой версии программного обеспечения)
3. Переносимость (мобильность, возможность многократного использования, функциональная совместимость – факторы качества важные для переносимости на другие аппаратные и программные платформы)



1. Модель Боэма

Эта модель имеет недостатки современных моделей, которые автоматически качественно проверяют работу. Модель Боэма пытается определить качественно показателей и метрик. Она показывает характеристики в более крупном масштабе, чем модель Маккола. В этой модели практичность описывает, как легко, надежно и эффективно может быть использован.

1. Модель Furps/furps +

Модель FURPS+

Модель FURPS+ это одна из классификаций требований. Используется в Rational Unified Process.

Аббревиатура FURPS составлена по первым буквам названий видов требований:

Functionality (функциональные требования)

Usability (требования к удобству работы)

Reliability (требования к надежности)

Performance (требования к производительности)

Supportability (требования к простоте поддержки)

Знак плюс "+" в аббревиатуре FURPS+ означает, что учитываются дополнительные виды требований:

* ограничения на структуру проекта
* требования к реализации
* требования к интерфейсу
* физические требования.

1. Модель Гецци

Модель качества программного обеспечения Гецци

Карло Гецци и соавторы различают качество продукта и процесса.

Характеристики ПО Гецци:

* целостность,
* надежность и устойчивость,
* производительность,
* практичность,
* верифицируемость,
* сопровождаемость,
* возможность многократного использования,
* мобильность,
* понятность,
* возможность взаимодействия,
* эффективность,
* своевременность реагирования,
* видимость процесса разработки

1. Модель качества Дроми

Устранение дефектов, которое должны быть устранено

1. Модель качества SATC

Обновление данных, когда нужно провести быструю надстройку, где дедлайн стоит уже через месяц

1. Модель качества ISO 9126

Критерий, где написано качество изначально при разработке, чтобы можно было сделать программу подстать выбранным требованиям

1. Модель качества QMOOD

Тестирование

Испытание программы – проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определенным образом.

ПМИ в соответствии с ГОСТ 19.301 должен включать следующие разделы:

1. Объект испытаний
2. Цель испытаний
3. Требования к программе
4. Требования к программной документации
5. Средства и порядок испытаний
6. Методы испытаний

Тестирование оказывает влияние на качество

Виды испытаний программных модулей:

1. Испытание компонент ПС – проверка работоспособности отдельных подсистем ПС. Проводятся только в исключительных случаях по специальному решению аттестационной комиссии. Работает вовремя разработки, тестирования, сборки и эксплуатации
2. Системные испытания – проверка работоспособности ПС в целом. Может включать те же виды тестирования, что и при комплексной откладке ПС.
3. Приемо-сдаточное испытание – основной вид испытаний при аттестации ПС.
4. Полевые испытания – демонстрация ПС вместе с технической системой, которой управляет эта ПС, узкому кругу заказчиков
5. Промышленный испытания – процесс передачи ПС в постоянную эксплуатацию пользователям.

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта

Оформление программной документации. Принципы разработки программного обеспечения.

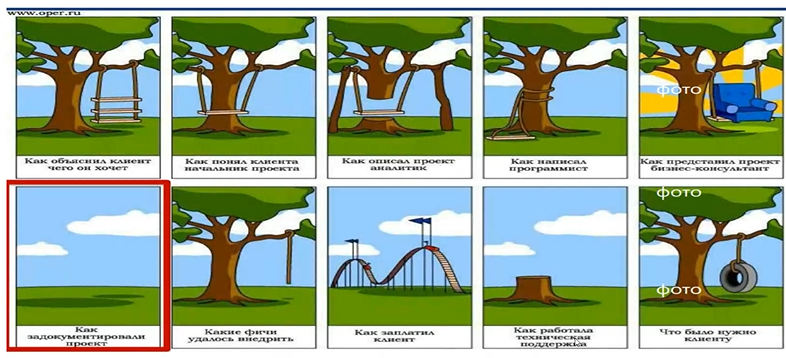
Программа – продукт

Для того, чтобы превратить программный код в продукт, нужно снабдить его всей необходимой документацией.

В рамках курсовой работы на ОП ПИ и проектной работы на ОП ПМИ создается продукт (программное обеспечение)

Продукт = программный код + документация

Программирование, алгоритмы и структуры данных, ООП и т.д. – учат писать код.



Международный стандарты

Список основных международных стандартов для написания документации:

IEEE Std 1063-2001 «IEEE Standard for Software User Documentation» - стандарт для написания руководство пользователя

IEE Std 1016-1998 «IEEE Recommended Practice»

Государственный стандарты разделены на 2 группы:

- ГОСТ 19 – Единая система программной документации (ЕСПД)

- ГОСТ 34 – Информационные технологии. Комплекс стандартов на автоматизированные системы

ГОСТ 19. ЕСПД

ЕСПД – комплекс государственных стандартов (ГОСТ), устанавливающих взаимосвязанные правила.

ГОСТ 34. ИТ

Автоматизированная система (АС) – система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию.

ГОСТ 19 VS ГОСТ 34

Автоматизированная система, как правило, содержит организационное решение под конкретно пользователя и заказчика.

Программа может быть создана и растиражирована под большое количество пользователей без привязки к какому-либо.