Лк №1
07.09.04

Архитектура и методы проектирования ПО

Argo UML
Rational Rose
следующая лекция - опрос по пункту 1.2 из лекции 1. прочитать все до конца файла.

Лк №2
07.09.10

Каскадная модель. в ее условиях разработка ПО осущ. по строго определенным этапам со строго определенными требованиями для каждого этапа и переход на след.этап осущ.только после выполнения всех задач текущего этапа.

Спиральная модель.

Инкрементальная модель. в ней каждая из итераций разработки ПО могут выполняться последовательно n-е кол-во раз. одна итерация предполагает выполнение следующих этапов: анализ требований, проектирование, реализация, компонентное тестирование, интеграция, тестирование целого продукта.

Унифицированный процесс разработки ПО (Unified software development process). использует язык UML. в его основе лежит разбиение процесма разработки на след.этапы:

1. начальная итерация (анализ)
2. проектирование (включая проектирование основных классов - чего в пред.методологиях не было)
3. итерация разработки (конструирования)
4. итерация выпуска готового продукта.

использование модели USDP оправдано в условиях сложных проектов, оперирующих с большим количеством данных, объектов и различных групп пользователей.

(для биллинговой системы оператора сотовой связи можно использовать именно эту методологию)

методология USDP подразумевает 6 моделей, которые связаны с моделями и диаграммами языка UML:

1. модель вариантов использования
2. аналитическая модель (описывает базовые классы приложения)
3. модель проектирования (описывает связи между классами и отдельными объектами)
4. модель развертывания ПО.
5. модель реализации исходного кода
6. модель тестирования ПО

еще одна методология - *экстремальное программирование*. основная его идея - в том, что требования к ПО могут поступать в совершенно произвольном порядке, в любое время и на любой стадии разработки ПО.

Прототипирование. эта методология предполагает, сто сначала разрабатыватся предварительная спецификация продукта, а затем на ее основе - модель будущего продукта. на основе этой модели разраб.окончательная спецификация и уже выпускается окончательная версия продукта.

Test-driven model (разработка, управляемая тестированием) - она предполагает, что контроль качеством в полном объеме осуществляется на каждом этапе разработки прогр.продукта. также предполагается, что на каждом этапе будут производиться какие-либо действия с точки зрения контроля качества.

методология MSF (Microsoft solution framework). если кратко, то это комбинация каскадной и спиральной модели. она обеспечивает упорядоченность каскадной модели и при этом предоставляет гибкость, свойственную спиральной модели. базовые принципы:

1. единое видение проекта (каждый разработчик в курсе всего, что происходит в проекте)
2. проявляйте гибкость (при изменении требований адекватно реагировать на это и применять соответствующие меры)
3. концентрируйтесь на бизнес-приоритетах (весь процесс разработки контроллируется приоритетами заказчика и вашего предприятия)
4. поощряйте свободное общение (обсуждение и обмен информацией между сотрудниками, которые работают над проектом, а также между руководителями проекта и заказчиком; разумно организовыват

конференции между заказчиком и ключевыми исполнителями проекта)

Лк №3
07.09.11

Проектирование, ориентированное на практичность

в практичный подход входят следующие ключевые элементы:

1. руководство по прагматичному проектированию
2. процесс проектирования, построенный на основе модели
3. организованные действия при разработке
4. итеративные улучшения
5. оценка качества

при создании практичного программного обеспечения ставится задача разработать такое ПО, которое будет наилучшим образом пониматься и использоваться пользователем. в центре внимания данного подхода находится работа, которую будут выполнять пользователи.

основные принципы проектирования, направленного на практичность:

1. модели и моделирование. при реализации данного принципа необходимо разработать модель пользовательского интерфейса будущей системы, провести его обсуждение и анализ с точки зрения удобства использования в будущем.
2. процессы разработки. при разработке программного обеспечения, учитывающего практичное проектирование, необходимо обеспечить такой цикл разработки ПО, который поможет решить следующие задачи: обеспечить масштабируемость выполняемого проекта, удовлетворение требованиям будущего проекта, контроль практичности на промежуточных стадиях разработки и демонстрация промежуточных версий будущим пользователям.
3. итеративное улучшение. практичный подход подразумевает, что система должна разрабатываться последовательно, с проведением промежуточных тестов, инспектированием кода и проверкой практичности ПО.
4. оценка качества. процесс проектирования, ориентированный на пользователя, должен сопровождаться всесторонним контролем качества, который включает следующие этапы: инспектирование исходного кода, тестирование функциональности, проверка возможности интеграции, проверка производительности и проверка удобства использования.

также существует принцип абстракции. его суть состоит в использовании абстрактных моделей для решения конкретных задач. т.е. вместо того, чтобы отображать буквально действия пользователей и конкретные объекты с которыми они работают, такие модели представляют абстракции, т.е. концепции и ключевые идеи, присущие данной задаче. когда идет речь об "абстракции", в первую очередь имеются в виду абстрактные типы данных и абстрактные классы.

архитектура интерфейса

целью всего процесса разработки пользовательского интерфейса является эффективный выбор и размещение визуальных элементов управления, целостность интерфейса (в пределах одного диалогового окна должны быть только те элементы управления, которые относятся именно к его главному предназначению, и не должно быть лишних элементов) и его интуитивная понятность.

понятие архитектуры пользовательского интерфейса относится к общей структуре интерфейса, а не к мелким его деталям. на этапе разработки архитектуры интерфейса не нужно уделять внимание таким вопросам как цвет интерфейса и элементов управления, картинки (иконки и т.п.), шрифты и прочие визуальные эффекты.

разработка архитектуры интерфейса может осуществляться с использованием графических редакторов, PowerPoint или же среды разработки ПО. *при разработке архитектуры пользовательского интерфейса в первую очередь следует уделить внимание только ключевым интерфейсам ПО.* при проектировании общей структуры интерфейса следует стремиться к тому, чтобы количество открывающихся и накладывающихся друг на друга окон было не больше трех. при этом стремиться, чтобы каждое последующее окно не перекрывало полностью предыдущее.

при проектировании практичного ПО и его интерфейса, желательно знать ответы на следующие вопросы:

1. кем являются будущие пользователи системы и как они с ней связаны.
2. какие задачи пользователи будут пытаться решать, работая с системой

3. что именно требуется от системы для решения задач пользователей, как она должна быть организована.
4. каковы условия эксплуатации системы
5. как должен выглядеть пользовательский интерфейс, каким должно быть его поведение.

при практичном проектировании используются следующие модели, помогающие найти ответы на первые три вопроса, указанные выше:
1. ролевая модель - модель взаимоотношений пользователя с системой
2. модель задач - это структура задач, которые необходимо решать пользователю
3. контентная модель - это модель информационного содержимого, которая отражает набор инструментов и материалов, предоставляемых пользовательским интерфейсом.
4. операционная модель - это операционный контекст использования системы, т.е. сценарий использования системы во времени.
5. модель реализации - визуальный проект интерфейса, описание его работы и исходный код системы.

все эти модели состоят из двух частей: набора описаний и карты связей между этими описаниями. всем этим моделям можно найти соответствия в языке UML.

Лк №4
07.09.17

Пользователь и среда в которой он работает

до начала разработки с-мы нужно задаться след.вопросами:
- чем должны заниматься будущие пользователи программы
- какие задачи они будут решать
- какие требования предъявл.к системе
- каким образом с-ма будет осущесвлять поддержку выполнения задач пользовтеля

в процессе разработки с-мы потребуется знать:
- что в системе не работает или работает неэффективно
- что нужно изменить в с-ме чтобы сделать ее более эффективной

для того, чтобы наилучшим образом представить будущего пользователя, предлагается использовать модель пользователя, построенную на основе предварительно собранной информации о будущих пользователях, их задачах и потребностях. такие модели пользователей можно строить с исп.обычных словесных описаний, стараясь аксимально учесть специфику и псизхологические аспекты рпеального пользователя. в дополнение к этому следует учесть интересы и потребности руководителей будущих пользователей и специалистов, кот.будут обучать будущих пользователей.

ориентироваться при разработке с-м на технически грамотных специалистов можно только в том случае, если с-ма будет экспоуатироваться только ними.

модели пользовательских ролей

пользовательская роль - это абстрактный набор потребностей, интересов, ожиданий, предполагаемого поведения и обязательств, характеризующая взаимоотношения между некоторы классом и с-мой.

основной процесс создания польз.роли строится на след.действиях: слушай, вникай, создавай модель, обсуждай ее с пользователем и вноси усовершенствования в модель.

с понятием роли очень тесно связано понятие актеров - это люди, которые выполняют соответствующую роль.

моделирование ролей. в простейшем случае модель роли - это совокупность требований, интересов, ожиданий и обязательств, характ.данную конкретную роль и отличающие ее от других. каждой роли желательно давать уникальное имя, фиксирующую осн.природу роли и идентифицирующее ее уникально. для каждой польз.роли справедливы вопросы, которые в начале лекции мы поставили для всей с-мы в целом. наилучшим подходом в выборе и создании ролей явл. подход, в котором сначала опред.базовые роли, а все остальные роли выстраиваются по принципу их значимости. при выявлении базовых ролей рекомендуется составлять список ролей-претендентов и на его основе уже определять базовые роли.

построение пользовательских ролей

1. раделение творческого процесса и критики. т.е разделить процесс разработки идей и их обсуждение. недопустимо критиковать идеи, которые еще окончательно не сформировались и оформились, в то же время подвеграть критике и обсуждению полностью оформленные идеи. все обсужденные и принятые идеи стараться хранить до тех пор, пока не закончится разработка проекта.
2. разделение общей перспективы и деталей.
3. разделение генерирования и организации.
4. накопление. нужно стараться накопить как можно более иссерпывающую информацию по всем пред.этапам.
5. организация. осуществить организацию всех накопленных деталей в единую структуру.
6. усовершенствование. после того как все готово, проверить и внести необходимые изменения.

карта пользовательских ролей - это инструмент, позвол.увидеть всю картину взаимоотношений пользователей между собой. внутри этой карты могут существовать след.типы польз.отнош.:

1. по сходству - исп. в том случае, если существует некое различимое подобие между ролями, но точно не установлена его природа.
2. классификационные - такие отношения возникают если польз.роль является подтипом другой польз.роли.
3. композиционные - объединяет в себе характеристики или свойства двух или более ролей.

Лк №5
07.09.18

Структурированные модели пользовательских ролей

структурированная модель ролей позволяет собирать и организовывать информацию о пользователях, направляя процесс развития целостной модели задач, выделяя для этого основные элементы use case и очерчивая грани *практического контекста* пользовательских ролей.

ограничения могут быть:

1. аппаратные
2. среда. существующие факторы физической среды, в которой находится реальный пользователь, выполняющий данную роль.
3. обязанности. следует учитывать специфику обязанностей будущего пользователя, который будет выполнять данную роль.
4. профиль обязанностей. представляет собой набор данных о реальных пользователях, которые будут работать в системе, и в первую очередь описывает их обязанности.
5. взаимодействие (профиль ролевых взаимодействий). содержит информацию о типичных или ожидаемых вариантах использования системы различными пользователями играющими различные роли. в этом профиле должна содержаться следующая информация:
   - частота использования данной роли
   - регулярность использования системы в целом
   - непрерывность взаимодействия пользователей с системой
   - концентрация. работа с системой распределена во времени или имеет импульсный характер.
   - интенсивность (какова скорость взаимодействия пользователя с системой).
   - сложность
   - предсказуемость (действия в определенных ролях достаточно предопределены)
   - местоположение управления. кем управляются взаимодействия - программой (процессом) или пользователем.

6. умения (профиль умений). описывает уровень квалификации и мастерства пользователей, которые будут работать с системой.
7. информация (профиль информации). необходимо определить источники, способы обмена и хранения информации по проекту и в частности по конкретной пользовательской роли. среди характеристик данного профиля следует отметить следующие:
   - источники исходящей инфомации (откуда берутся данные - от конечного пользователя, от руководителя или может от экспертов...)
   - направление потока информации от/к пользователю (какое направление потока данных доминирует - от пользователя или к пользователю).

8. критерии практичности. они описываются профилем практичности, который содержит следующие

характеристики:
- функциональная поддержка
- удобство работы с системой
- надежность системы
- соответствие требованиям производительности

на след.раз выписать свое видение функций системы, по своему заданию.
конечный объем всего задания - не больше 20 страниц (со всеми диаграммами и т.п.). можно 10,8..

Лк №6
07.09.25

Концепции UML, представления и диаграммы

В основе языка UML лежат представления и диаграммы.
представление - это подмножество конструкций, которое представляет один из аспектов моделируемой системы.
концепции каждого из представлений проиллюстрированы одной или несколькими диаграммами.
на самых высоких уровнях абстракции различают следующие группы представлений:
- структурная классификация
- динамическое поведение
- физическое размещение
- управление моделью

структурная группа представлений описывается следующими диаграммами:
1. диаграмма классов
2. диаграмма внутренней структуры
3. диаграмма кооперации
4. диаграмма компонентов
5. диаграмма use case

динамическая группа представлений:
1. представление конечных автоматов, описываемое диаграммой конечных автоматов
2. представление деятельности, описываемое диаграммой деятельности
3. представление взаимодействия, описываемое диаграммой последовательностей и диаграммой коммуникаций

физическая группа представлена одним представлением развертывания, которое представляется диаграммой развертывания.

управление моделью представлено :
- представлением управления моделью, которое описывается диаграммой пакетов
- представлением профиля, которое тоже описывается диаграммой пакетов (отличие от представления управления моделью заключается в том, что в диаграмме используются разные наборы графических элементов)

11. разработать систему учета оплаты услуг для интернет-провайдера

система будет работать на одном сервере (*nix), также имеется БД MySQL. логи затраченного трафика клиентов пишутся в определенную папку. для быстрой обработки логов, соответствующая часть системы должна быть написана на C/C++. web-интерфейс может быть написан на php. необходимо написать систему, которая будет выполнять следующие функции:
- через определенные промежутки времени (допустим, каждый час) обрабатывать логи, занося обработанную информацию о затраченном трафике и деньгах в БД. затраты по деньгам рассчитываются в зависимости от тарифного плана клиента и "зон" ip-адресов (локальный трафик - бесплатно, харьковский трафик, мир).
- каждый день клиентам должны отправляться отчеты о затраченном трафике и остатке на счету
- должен быть web-интерфейс, позволяющий каждому клиенту просмотреть подробную информацию о затраченном трафике и деньгам, по дням.
- также должен быть web-интерфейс для администраторов, позволяющий изменить тарифный план

каждого клиента, просмотреть статистику, добавить денег на счет клиента при оплате клиентом услуг.

- когда затраченная сумма денег превышает остаток на счету, система должна блокировать клиента, позволяя ему лишь просмотреть статистику.

Лк №7
07.10.01

Модели задач и их представление моделями use case

самая простая модель задачи - это блок-схема алгоритма. более сложная вариация модели - иерархическая модель. эти модели осуществляют декомпозицию задачи на четкий набор подзадач и выстраивают эти подзадачи в определенной иерархии. третья модель задачи - это сценарий. сценарии очень тесно связаны с use case.

элемент use case - это ситуация, вариант использования или применения какого-то элемента системы. use case предназначен для того, чтобы описать следующие сущности:

- обеспечение функциональности
- внешняя точка зрения
- повествовательное описание

- описание роли
- описание взаимодействия между пользователем и системой
- завершенное и понятное пользователю применение системы

use case - это полноправная модель задачи, которая описывает саму задачу, но может также служить и связующим элементом между ролями, сценариями и вариантами использования системы.

сущностные элементы use case - включают в себя модель пользовательских устремлений и модель обязательств системы.

уровни абстракции и обобщения при разработке архитектуры системы

уровень абстракции документации по проекту должен соответствовать уровню разработчиков проекта.

уровень универсализации системы (обобщения) должен быть ровно таким, насколько он обеспечивает выполнение задач пользователям.

объединение элементов use case в какую-либо диаграмму или схему называется картой элементов use case, а в терминологии языка UML, это диаграмма use case.

Лк №8
07.10.02

Разработка пользовательского интерфейса

Задача построения хорошего пользовательского интерфейса включает в себя определение инструментов, материалов (и хранилищ для них), распределение содержимого интерфейса по множеству разных, но взаимосвязанных пространств взаимодействия.

модель содержимого интерфейса (модель содержимого) - это абстрактный образ содержимого различных пространств взаимодействия, входящих в состав системы, а также связи между ними.

пространство взаимодействия - это некоторая часть пользовательского интерфейса, представленная окном, экраном, диалоговым окном, страницей диалога с закладками.

карта навигации - это набор связей между различными пространствами взаимодействия.

общие требования к проектированию пользовательского интерфейса:

1. на основе требований к системе, архитектуре и диаграмм use case, определить пространство взаимодействия. при этом обеспечить оптимальное распределение функций системы между различными пространствами взаимодействия.
2. обеспечить выполнение связанных между собой функций в пределах одного пространства взаимодействия.
3. обеспечить оптимальную карту навигации.
4. обеспечить понятный контент для каждого пространства взаимодействия.
5. обеспечить цветовую гамму всех пространств взаимодействия таким образом, чтобы она не отвлекала пользователя от выполнения его задач.
6. обеспечить интуитивную понятность каждого пространства взаимодействия за счет использования иконок, пиктограмм и всплывающих подсказок.

контент - это информационное наполнение каждого пространства взаимодействия, состоящее из текстов возле элементов управления, надписей на элементах управления и всплывающих подсказок, а также самих элементов управления.

при проектировании карты навигации, необходимо обеспечить, чтобы вложенность вызываемых окон была не более трех.

прототипирование интерфейса предполагает проработку всего интерфейса приложения с использованием среды разработки или с использованием power point, или с использованием бумаги и карандаша.

разработка визуального интерфейса.

существуют два подхода к разработке интерфейса:

1. разработка красивого и цветастого интерфейса
2. разработка утилитарного интерфейса (с которым удобно работать). при этом такой интерфейс тоже может быть симпатичным.

утилитарный интерфейс - это удобный в работе, простой в понимании, с хорошо подобранной цветовой гаммой и хорошим набором пиктограмм.

существует 3 канала коммуникации с пользователем:

1. текстовый
2. графический
3. звуковой

общие правила использования каналов связи:

- при написании слов, использовать единый шрифт во всей программе
- выбирать такой шрифт, чтобы он легко читался
- разрабатывать иконки и пиктограммы таким образом, чтобы они были запоминающимися и отражали суть той функциональности, к которой они относятся.

исследованиями установлено, что наилучшее восприятие информации происходит, когда графические и текстовые каналы связи совмещены вместе.

- ни в коем случае не использовать на интерфейсе в качестве надписей к элементам управления какие-либо нестандартные и забавные шрифты.
- не размещать на интерфейсе большие красочные изображения, которые не несут никакой практической нагрузки для пользователя.
- при использовании звуков, обязательно дать возможность пользователю отключать их.
- стараться использовать графические и текстовые каналы связи сбалансированно, а в таких элементах управления как меню и кнопки - в обязательном порядке включать в графический элемент и текст.
- при размещении элементов управления в окне, стремиться разместить их равномерно по всей области, объединить элементы управления в группы по сходному их смыслу и обеспечить удобный переход между элементами управления.
- выбрать цветовую гамму интерфейса таким образом, чтобы не раздражать пользователя и не переутомлять его зрение.
- размещать элементы управления интерфейса таким образом, чтобы перемещения мышки от одного элемента к другому было по возможности минимальным.
- обеспечить возможность использования клавиатуры во всех пространствах взаимодействия (во всем приложении).

Лк №9
07.10.15

разработка справочных систем

при разработке справочной системы необходимо предоставить пользователю возможность найти ответы на следующие вопросы:

<table>
<tr><td rowspan="2">вопрос пользователя</td><td colspan="2"></td><td rowspan="2"></td></tr>
<tr><td></td><td></td></tr>
<tr><td>что это</td><td></td><td></td><td></td></tr>
</table>

(таблица с фотки)

в руководстве пользователя желательно отобразить следующие моменты:

1. описание всей функциональности системы
2. описание возможных сценариев решения конкретных пользовательских задач ("how to")
3. устранение возможных проблем
4. установка и настройка системы

для проверки полноты и корректности справки, рекомендуется дать программу каким-либо специалистам незнакомым с ней, и попросить их научиться работать с программой с использованием справки.

настоятельно рекомендуется использовать всплывающие подсказки для абсолютно всех элементов управления в программе.

также рекомендуется использовать контекстную справку.

Лк №10
07.10.29

Типичные компоненты архитектуры программного продукта и типичные требования к ПО:

1. организация программы
2. основные классы системы
3. организация данных
4. бизнес-правила
5. пользовательский интерфейс
6. управление ресурсами
7. безопасность
8. производительность
9. масштабируемость
10. взаимодействие с другими системами (интеграция)
11. интернационализация, локализация
12. ввод-вывод данных
13. обработка ошибок

Лк №11
07.11.12

контрольный список вопросов, при помощи которого можно проверить качество архитектуры:

1. ясно ли описана общая организация программы, включает ли спецификация обзор архитектуры и ее обоснование
2. адекватно ли определены основные компоненты программы, их области ответственности и взаимодействия с другими компонентами
3. все ли функции, указанные в спецификации требований, реализованы разумным количеством компонентов системы
4. приведено ли описание самых важных классов и их обоснование
5. приведено ли описание организации базы данных
6. определены ли все бизнес-правила и описано ли их влияние на систему
7. описана ли стратегия проектирования пользовательского интерфейса
8. сделан ли пользовательский интерфейс модульным, чтобы его изменения не влияли на оставшуюся часть системы
9. приведено ли описание стратегии ввода-вывода данных
10. проведен ли анализ производительности системы, которая будет реализовываться с использованием данной архитектуры
11. проведен ли анализ надежности проектируемой системы
12. проведен ли анализ вопросов масштабируемости и расширяемости системы

методики проектирования:

1. методика итераций
2. нисходящий и восходящий подходы к проектированию
3. экспериментальное проектирование
4. совместное проектирование

рекомендуется составить таблицу рисков проекта:

№	компоненты системы	уровень сложности каждого компонента	риск разработки данного компонента
1.		средний	средний
2.			

качество программного обеспечения

качество - это удовлетворение требованиям заказчика

характеристики качества ПО:

1. корректность - отсутствие дефектов в спецификации и реализации системы
2. практичность - легкость изучения и использования системы
3. эффективность - степень использования системных ресурсов
4. надежность - способность системы выполнять заданные функции в определенных условиях

5. целостность - способность системы предотвращать несанкционированный или некорректный доступ к своим программам и данным

6. адаптируемость - возможность использования системы без ее изменения в различных областях для решения различных задач

7. правильность - степень безошибочности системы, особенно в отношении вывода количественных данных

8. живучесть - способность продолжать работу при вводе некорректных данных

9. удобство сопровождения - легкость изменения програмной системы с целью реализации дополнительных возможностей

10. гибкость - возможность изменения системы для применения в различных областях

11. портируемость - легкость адаптации системы к новым средам, на которые она не была ориентирована изначально. возможность повторного использования компонентов системы в других системах

12. масштабируемость

13. простота исходного кода - легкость чтения и понимания исходного кода системы

14. тестируемость - наличие возможностей по проверке системы и наличие четких сценариев по проверке системы.

15. понятливость - легкость понимания системы на уровне организации и уровне исходного кода

контроль качества - это набор мероприятий, направленный на снижение количества дефектов в системе и на повышение надежности и отказоустойчивости системы

Лк №12
07.11.26

10.12 - последние лабы. прийти всем и сделать все по заданию.

методики повышения качества ПО
