Лекция 3. Эксплуатационные требования к программным продуктам. Разработка технического задания

Эксплуатационные требования определяют некоторые характеристики разрабатываемого ПО, проявляемые в процессе его функционирования. К таким характеристикам относят:

• правильность – функционирование в соответствии с техническим заданием;

• универсальность – обеспечение правильной работы при любых допустимых данных и защиты от неправильных данных;

• надежность (помехозащищенность) – обеспечение полной повторяемости результатов, т.е. обеспечение их правильности при наличии различного рода сбоев;

• проверяемость – возможность проверки получаемых результатов;

• точность результатов – обеспечение погрешности результатов не выше заданной;

• защищенность – обеспечение конфиденциальности информации;

• программная совместимость – возможность совместного функционирования с другим ПО;

• аппаратная совместимость – возможность совместного функционирования с оборудованием;

• эффективность – использование минимально возможного количества ресурсов технических средств (времени микропроцессора, объема оперативной памяти);

• адаптируемость – возможность быстрой модификации с целью приспособления к изменяющимся условиям функционирования;

• повторная входимость – возможность повторного выполнения без перезагрузки с диска;

• реентерабельность – возможность «параллельного» использования несколькими процессами.

Правильность - обязательное требование для любого ПО: все, что указано в ТЗ, должно быть реализовано. Следует понимать, что ни тестирование, ни верификация не доказывают правильности созданного программного продукта. Поэтому говорят об определенной *вероятности наличия ошибок.* Чем большая ответственность перекладывается на компьютерную систему, тем меньше должна быть вероятность программного и аппаратного сбоя. Например, вероятность неправильной работы для системы управления атомной электростанцией должна быть близка к нулю.

Универсальность - также входит в группу обязательных требований. Ничего хорошего нет, если разработанная система выдает результат для некорректных данных или аварийно завершает свою работу на некоторых наборах данных. Но доказать универсальность программы, как и ее правильность, невозможно, поэтому говорят о степени универсальности программы.

Чем выше требования к правильности и универсальности ПО, тем выше и требования к его *надежности.* Источниками помех могут являться все участники вычислительного процесса: технические и программные средства, люди. Технические средства подвержены сбоям (из-за резких скачков напряжения питания или помех при передаче информации по сетям). ПО может содержать ошибки. Люди могут ошибаться при вводе исходных данных.

Современные вычислительные устройства достаточно надежны. Сбои технических средств регистрируются аппаратно, результаты вычислений восстанавливаются. При длительных вычислениях промежуточные результаты сохраняют (создание контрольных точек), что позволяет при сбое продолжить вычисления с данными, записанными в последней контрольной точке.

Передача информации по сетям аппаратно контролируется, применяется помехозащитное кодирование, которое позволяет находить и исправлять ошибки передачи данных. Однако полностью исключить ошибки технических средств невозможно, поэтому в случаях, когда требования к надежности высоки, используют дублирование систем, при котором две системы решают одну и ту же задачу параллельно, периодически сверяя полученные результаты.

Часто самым «ненадежным элементом» современных систем являются люди. В условиях монотонной работы операторы допускают большое количество ошибок. Известны средства, позволяющие снизить количество ошибок в конкретных ситуациях. Там, где это возможно, используют ввод избыточной информации, позволяющий выполнять проверки правильности вводимых данных (ввод контрольных сумм и т.п.). Широко используют подсказки, когда информацию необходимо не вводить, а выбирать из некоторого списка и т.п.

Повышенные требования к надежности предъявляют при разработке систем управления, функционирующих в режиме реального времени, когда вычисления выполняются параллельно с технологическими процессами. Это требование важно и для научно-технических систем и БД.

Для обеспечения *проверяемости* следует документально фиксировать исходные данные, установленные режимы и прочую информацию, которая влияет на получаемые результаты. Это существенно в случаях, когда данные поступают непосредственно от датчиков.

Точность или величина погрешности результатов зависит от точности исходных данных, степени адекватности используемой модели, точности выбранного метода и погрешности выполнения операций в компьютере. Требования к точности результатов наиболее жесткие для систем управления технологическими процессами (например, химическими) и систем навигации (например, система управления стыковкой космических аппаратов).

Обеспечение *защищенности* (конфиденциальности) информации - отдельная и в условиях наличия сетей достаточно сложная задача. Помимо чисто программных средств защиты (кодирование информации, идентификация пользователя) используют специальные организационные приемы. Наиболее жесткие требования предъявляются к системам, в которых хранится информация, связанная с государственной и коммерческой тайной.

Требование *программной совместимости* может варьироваться от возможности совместной установки с указанным ПО до обеспечения взаимодействия с ним (обмена данными и т.п.). Часто приходится обеспечивать функционирование ПО под управлением заданной ОС. Однако может потребоваться предусмотреть получение данных из программы или передачу некоторых данных ей. Тогда необходимо точно оговорить форматы передаваемых данных.

Требование *аппаратной совместимости* формулируют в виде минимально возможной конфигурации оборудования, на котором будет работать ПО. При использовании нестандартного оборудования должны быть указаны интерфейсы или протоколы обмена информацией.

Эффективность системы обычно оценивается отдельно по каждому ресурсу вычислительной установки. Часто используют критерии:

• время ответа системы – для систем, взаимодействующих с пользователем в интерактивном режиме, и систем реального времени;

• объем оперативной памяти – для продуктов, работающих в системах с ограниченным объемом ОП;

• объем внешней памяти – для продуктов, использующих внешнюю память (БД);

• количество обслуживаемых внешних устройств – для продуктов, осуществляющих интенсивное взаимодействие с внешними устройствами, например датчиками.

Требования эффективности могут противоречить друг другу: чтобы уменьшить время выполнения фрагмента программы, может потребоваться дополнительный объем ОП.

Адаптируемость - оценивает технологическое качество ПО, поэтому оценить эту характеристику количественно практически невозможно. Можно только констатировать, что при создании продукта использованы технологии и приемы, облегчающие его модернизацию.

Требование *повторной входимости* предъявляется к ПО, резидентно загруженному в ОП (драйверам). Для обеспечения данного требования необходимо так организовать программу, чтобы никакие её исходные данные не затирались в процессе выполнения или восстанавливались в начале или при завершении каждого вызова.

Требование *реентерабельности* - более жесткое, чем повторная входимость, в этом случае все данные, изменяемые программой в процессе выполнения, должны быть выделены в специальный блок, копия которого создается для каждого процесса при вызове программы.

Сложность многих программных систем не позволяет сразу сформулировать четкие требования к ним. Обычно для перехода от идеи создания некоторого ПО к четкой формулировке требований, которые могут быть занесены в техническое задание, необходимо выполнить предпроектные исследования в области разработки.

Предпроектные исследования предметной области

Цель предпроектных исследований - преобразование общих нечетких знаний о предназначении будущего ПО в сравнительно точные требования к нему.

Существуют два варианта неопределенности:

• неизвестны методы решения формулируемой задачи – такого типа неопределенности обычно возникают при решении научно-технических задач;

• неизвестна структура автоматизируемых информационных процессов – обычно встречается при построении автоматизированных систем управления предприятиями.

В первом случае во время предпроектных исследований определяют возможность решения поставленной задачи и методы, позволяющие получить требуемый результат, что может потребовать соответствующих научных исследований фундаментального и прикладного характера, разработки и исследования новых моделей объектов реального мира.

Во втором случае определяют:

• структуру и взаимосвязи автоматизируемых информационных процессов;

• распределение функций между человеком и системой, между аппаратурой и ПО;

• функции ПО; внешние условия его функционирования, особенности интерфейсов;

• требования к программным и информационным компонентам, необходимые аппаратные ресурсы, требования к БД и физические характеристики программных компонент.

Результаты предпроектных исследований предметной области используют в процессе разработки технического задания (ТЗ).

Разработка технического задания

Техническое задание – это документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемно-сдаточных испытаний. В разработке ТЗ участвуют представители заказчика и исполнителя. В основе этого документа лежат исходные требования заказчика, результаты выполнения научно-исследовательских работ, исследований, научного прогнозирования и т.п.

Основные факторы, определяющие характеристики разрабатываемого ПО:

• исходные данные и требуемые результаты, которые определяют *функции* программы;

• среда функционирования (программная и аппаратная) – может быть задана, а может выбираться для обеспечения параметров, указанных в ТЗ;

• возможное взаимодействие с другим ПО и/или специальными техническими средствами – также может быть определено, а может выбираться исходя из набора выполняемых функций.

Разработка ТЗ выполняется в следующей последовательности. Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Затем определяют перечень результатов, их характеристики и способы представления.

Далее уточняют среду функционирования ПО: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, версии и параметры другого установленного ПО, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое ПО собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

На ТЗ существует **стандарт ГОСТ 19.201-78** «Техническое задание. Требования к содержанию и оформлению». В соответствии со этим стандартом ТЗ должно содержать разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки.

При необходимости допускается в техническое задание включать приложения.

Введение должно включать наименование и краткую характеристику области применения программного продукта, а также объекта (например, системы) в котором предполагается их использовать. Основное назначение введения – продемонстрировать актуальность данной разработки и показать, какое место эта разработка занимает в ряду подобных.

Раздел *Основания для разработки* должен содержать наименование документа, на основании которого ведется разработка, организации, утвердившей данный документ, и наименование или условное обозначение темы разработки (план, приказ, договор и т.п.).

Раздел *Назначение разработки* должен содержать описание функционального и эксплуатационного назначения программного продукта с указанием категорий пользователей.

Раздел *Требования к программе или программному изделию* должен включать подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

Наиболее важный - подраздел *Требования функциональным характеристикам.* В нем должны быть перечислены выполняемые функции и описаны состав, характеристики и формы представления исходных данных и результатов. При необходимости указывают критерии эффективности: максимально допустимое время ответа системы, максимальный объем используемой оперативной и/или внешней памяти и др. Если разработанное ПО не будет выполнять указанных в ТЗ требований, то оно считается не соответствующим ТЗ, т.е.

неправильным с точки зрения критериев качества. Универсальность будущего продукта специально не оговаривается, но подразумевается.

В подразделе *Требования к надежности* указывают уровень надежности, который должен быть обеспечен разрабатываемой системой и время восстановления системы после сбоя. Для систем с обычными требованиями к надежности в этом разделе иногда регламентируют действия разрабатываемого продукта по увеличению надежности результатов (контроль входной и выходной информации, создание резервных копий промежуточных результатов и т.п.).

В подразделе *Условия эксплуатации,* указывают особые требования к условиям эксплуатации: температуре окружающей среды, относительной влажности воздуха и т.п. Подобные требования формулируют, если разрабатываемая система будет эксплуатироваться в нестандартных условиях или использует специальные внешние устройства, например для хранения информации. Здесь указывают вид обслуживания, необходимое количество и квалификация персонала. Иначе допускается указывать, что требования не предъявляются.

В подразделе *Требования к составу и параметрам технических средств* указывают необходимый состав технических средств с указанием их основных технических характеристик: тип микропроцессора, объем памяти, наличие внешних устройств и т.п. При этом часто указывают два варианта конфигурации: минимальный и рекомендуемый.

В подразделе *Требования к информационной и программной совместимости* можно задать методы решения, определить язык или среду программирования для разработки, используемую операционную систему и другие системные и пользовательские программные средства, с которым должно взаимодействовать разрабатываемое ПО. В этом разделе при необходимости указывают, какую степень защиты информации необходимо предусмотреть.

В разделе *Требования к программной документации* указывают необходимость наличия руководства программиста, руководства пользователя, руководства системного программиста, пояснительной записки и т.п. На все эти типы документов существуют ГОСТы.

В разделе *Технико-экономические показатели* рекомендуется указывать ориентировочную экономическую эффективность, предполагаемую годовую потребность и экономические преимущества по сравнению с существующими аналогами.

В разделе *Стадии и этапы разработки* указывают стадии разработки, этапы и содержание работ с указанием сроков разработки и исполнителей.

В разделе *Порядок контроля и приемки* указывают виды испытаний и общие требования к приемке работы.

В приложениях при необходимости приводят: перечень научно-исследовательских работ, обосновывающих разработку; схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые следует использовать при разработке.

В зависимости от особенностей разрабатываемого продукта разрешается уточнять содержание разделов: использовать подразделы, вводить новые разделы или объединять их.

В случаях, если какие-либо требования, предусмотренные ТЗ, заказчик не предъявляет, в соответствующем месте следует указать «Требования не предъявляются».

Разработка ТЗ – процесс трудоемкий. Наиболее сложное - четкое формулирование основных разделов: введения, назначения и требований к программному продукту.

В качестве примеров рассмотрим два ТЗ.

Пример 1. Разработать ТЗ на программный продукт, предназначенный для наглядной демонстрации графиков функций одного аргумента y = f(x). Программа должна рассчитывать таблицу значений и строить график функций на заданном отрезке по заданной формуле и менять шаг аргумента и границы отрезка. Программа должна запоминать введенные формулы.

1.ВВЕДЕНИЕ

Настоящее ТЗ распространяется на разработку программы построения графиков и таблиц значений функций одной переменной, предназначенной для использования школьниками.

В школьном курсе элементарной алгебры тема анализа функций является одной из самых сложных. При изучении данной темы школьники должны научиться исследовать и строить графики функций одной переменной, используя известные характеристические точки функции, включая корни, точки разрыва первого и второго рода и т.д.

Существующее ПО, которое может решать подобные задачи, является универсальным, например MathCad. Оно имеет сравнительно сложный пользовательский интерфейс, ориентированный на пользователя, прослушавшего институтский курс высшей математики, что делает использование подобных средств школьниками невозможным. Разрабатываемая программа позволит школьникам проверить свои знания при изучении указанной темы.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Программа разрабатывается в соответствии с договором от ….

3. НАЗНАЧЕНИЕ

Основным назначением программы является помощь школьникам при изучении раздела «Исследование функций одного аргумента» школьного курса элементарной алгебры.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1.Требования к функциональным характеристикам

4.1.1. Программа должна обеспечивать возможность выполнения следующих функций:

- ввод аналитического представления функции одной переменной и хранение его;
- ввод и изменение интервала определения функции;
- ввод и корректировку шага аргумента;

• построение таблицы значений функции или изображение графика функции на заданном интервале при условии, что на указанном интервале она не имеет точек разрыва.

4.1.2. Исходные данные:

• аналитическое задание функции;

• интервал определения функции;

• шаг изменения аргумента, определяющий количество точек на интервале.

4.2. Требования к надежности

4.2.1.Предусмотреть контроль вводимой информации.

4.2.2.Предусмотреть блокировку некорректных действий пользователя при работе с системой.

4.3. Требования к составу и параметрам технических средств

4.3.1.Система должна работать на IBM совместимых персональных компьютерах.

4.3.2.Минимальная конфигурация:

• тип процессора...;

• объем оперативного запоминающего устройства.....

4.4. Требования к информационной и программной совместимости

Система должна работать под управлением семейства операционных систем ….

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1. Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

5.2.Разрабатываемая программа должна включать справочную информацию об основных терминах соответствующего раздела математики и подсказки учащимся.

5.3.В состав сопровождающей документации должны входить:

5.3.1.Пояснительная записка на 25-30 листах, содержащая описание разработки.

5.4.Руководство пользователя.

Пример 2. Разработать ТЗ на создание системы «Учет успеваемости студентов». Система предназначена для оперативного учета успеваемости студентов в сессию сотрудниками деканата. Сведения об успеваемости студентов должны храниться в течение всего срока их обучения и использоваться при составлении справок о прослушанных курсах и приложений к диплому.

1. ВВЕДЕНИЕ

Настоящее ТЗ распространяется на разработку системы учета успеваемости студентов, предназначенной для сбора и хранения информации о ходе сдачи экзаменационной сессии. Предполагается, что использовать данную систему будут сотрудники деканата.

Во время сессии необходимо получение оперативной информации о ходе ее сдачи студентами, однако выполнение такого контроля вручную требует значительного времени.

Автоматизированная система учета успеваемости позволит улучшить качество контроля сдачи сессии со стороны куратора и деканата и обеспечит получение сведений о динамике работы каждого студента, группы и курса в целом. Хранение информации о сдаче сессий в течение всего времени обучения позволит осуществлять автоматическую генерацию справок о прослушанных курсах и приложений к диплому выпускника.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Система разрабатывается на основании приказа декана № ... от ... и в соответствии с планом мероприятий по совершенствованию учебного процесса на … учебный год.

3. НАЗНАЧЕНИЕ

Система предназначена для хранения и обработки сведений об успеваемости студентов учебных групп факультета в течение всего срока обучения. Обработанные сведения об успеваемости студентов могут быть использованы для оценки успеваемости каждого студента, группы, курса и факультета в целом.

4. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1. Требования к функциональным характеристикам

4.1.1.Система должна обеспечивать возможность выполнения следующих функций:

- инициализацию системы (ввод списков групп, перечней изучаемых дисциплин в соответствии с учебными планами и т.п.);
- ввод и коррекцию текущей информации о ходе сдачи сессии студентами;
- хранение информации об успеваемости в течение времени обучения студента;
- получение сведений о текущем состоянии сдачи сессии студентами.

4.1.2. Исходные данные:

- списки студентов учебных групп;
- учебные планы кафедр - перечень предметов и контрольных мероприятий по каждому предмету;
- расписания сессий;
- текущие сведения о сдаче сессии каждым студентом.

4.1.3. Результаты:

- итоги сдачи сессии конкретным студентом;
- итоги сдачи сессии студентами конкретной группы;
- процент успеваемости по всем студентам группы при сдаче конкретного предмета в целом на текущий момент;
- проценты успеваемости по всем группам специальности на текущий момент;
- проценты успеваемости по всем группам курса на текущий момент;
- проценты успеваемости по всем курсам и по факультету на текущий момент;
- список задолжников группы и курса на текущий момент.

4.2. Требования к надежности

4.2.1.Предусмотреть контроль вводимой информации.

4.2.2.Предусмотреть блокировку некорректных действий пользователя.

4.2.3.Обеспечить целостность хранимой информации.

4.3. Требования к составу и параметрам технических средств

4.3.1.Система должна работать на IBM совместимых персональных компьютерах.

4.3.2.Минимальная конфигурация:

• тип процессора ....;

• объем оперативного запоминающего устройства.....

4.4. Требования к информационной и программной совместимости

Система должна работать под управлением семейства операционных систем ….

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1.Разрабатываемые программные модули должны быть самодокументированы, тексты программ должны содержать необходимые комментарии.

5.2. Программная система должна включать справочную информацию о работе и подсказки пользователю.

5.3. В состав сопровождающей документации должны входить:

5.3.1. Пояснительная записка на 25-30 листах, содержащая описание разработки.

5.3.2. Руководство системного программиста.

5.3.3. Руководство пользователя.

5.3.4. Графическая часть на трех листах формата А1;

1.3.4.1. Схема структурная программной системы.

1.3.4.2. Диаграмма компонентов данных.

1.3.4.3. Формы интерфейса пользователя.

Принципиальные решения начальных этапов проектирования

На начальных этапах процесса проектирования должны быть приняты принципиальные решения, во многом определяющие этот процесс, а также качество и трудоемкость разработки:

• выбор архитектуры ПО;

• выбор типа пользовательского интерфейса и технологии работы с документами;

• выбор подхода к разработке (структурного или объектного);

• выбор языка и среды программирования.

Эти решения определяют, что проектируется, с какими потребительскими характеристиками, как и какими средствами. Часть решений может быть определена в ТЗ, образовав группу *технологических требований,* остальные должны быть приняты как можно раньше, так как представляют собой исходные данные для процесса проектирования.

Выбор архитектуры программного обеспечения. *Архитектура ПО* - совокупность базовых концепций (принципов) его построения. Архитектура ПО определяется сложностью решаемых задач, степенью универсальности разрабатываемого ПО и числом одновременно работающих пользователей. Различают архитектуры:

• однопользовательскую – ПО рассчитано на одного пользователя, работающего за ПК;

• многопользовательскую – ПО рассчитано на работу в локальной или глобальной сети. Реализуют системы, построенные по принципу «клиент-сервер»

В рамках однопользовательской архитектуры различают: программы; пакеты программ; программные комплексы; программные системы.

Программа - адресованный компьютеру набор инструкций, точно описывающий последовательность действий, которые необходимо выполнить *для решения конкретной задачи.* При структурном подходе программы представляют собой иерархию подпрограмм, вызывающих друг друга в процессе решения поставленной задачи, при объектном подходе – совокупность обменивающихся сообщениями объектов, для реализации которых разработаны специальные классы. Программа в этом случае представляет собой отдельно компилируемую программную единицу, которая может использовать стандартные библиотеки подпрограмм, но, как правило, не организует свои. Это самый простой вид архитектуры, для решения небольших задач.

Пакеты программ - *совокупность* программ, решающих *задачи некоторой прикладной области.* Например, пакет графических или математических программ. Программы пакета связаны между собой только принадлежностью к определенной прикладной области. Пакет программ реализуют как набор отдельных программ, каждая из которых сама вводит необходимые данные и выводит результаты. По сути, пакет программ - это некоторая *библиотека программ.*

Программные комплексы - совокупность программ, *совместно* обеспечивающих решение *небольшого класса сложных задач одной прикладной области. Для* решения задачи может потребоваться решить несколько подзадач, последовательно вызывая программы комплекса. Вызов программ в программном комплексе осуществляется специальной программой – *диспетчером,* который обеспечивает несложный интерфейс с пользователем и, возможно, выдачу некоторой справочной информации. От пакета программ программный комплекс отличается еще и тем, что несколько программ могут последовательно или циклически вызываться для решения одной задачи, желательно хранить исходные данные и результаты вызовов в пределах одного пользовательского проекта. Программы в этом случае могут реализовываться отдельно и как совместно компилируемые программные единицы, а исходные данные храниться в ОП, в файлах.

Программные системы представляют собой *организованную совокупность* программ *(подсистем),* позволяющую решать *широкий класс задач из некоторой прикладной области.* В отличие от программных комплексов программы, входящие в программную систему,

взаимодействуют через обилие данные. Программные системы обычно имеют развитые пользовательский и внутренние интерфейсы, что требует их тщательного проектирования.

Многопользовательские программные системы в отличие от обычных программных систем должны организовывать *сетевое взаимодействие* отдельных компонентов ПО, что еще усложняет процесс его разработки. Для разработки подобного ПО используют специальные технологии или платформы (CORBA, COM, Java и т.п.).

Выбор типа пользовательского интерфейса (ПИ). Различают типы ПИ:

• *примитивные* – реализуют единственный сценарий работы, например, ввод данных – обработка – вывод результатов;

• *меню* – реализуют множество сценариев работы, операции которых организованы в иерархические структуры, например, «вставка»: «вставка файла», «вставка символа» и т.д.;

• *со свободной навигацией* – реализуют множество сценариев, операции которых не привязаны к уровням иерархии, и предполагают определение множества возможных операций на конкретном шаге работы; такие интерфейсы в основном используют Windows-приложения;

• *прямого манипулирования* – реализуют множество сценариев, представленных в операциях над объектами, основные операции инициируются перемещением пиктограмм объектов мышью.

Тип ПИ во многом определяет сложность и трудоемкость разработки..

Появление объектно-ориентированных визуальных сред разработки ПО, использующих событийный подход к программированию и рассчитанных на создание интерфейсов со свободной навигацией, существенно снизило трудоемкость разработки подобных интерфейсов.

Выбор типа ПИ включает и выбор *технологии работы с документами.*

Различают технологии:

• однодокументная - однодокументный интерфейс (SDI – Single Document Interface);

• многодокументная - многодокументный интерфейс (MDI– Multiple Document Interface).

Многодокументную технологию используют, если ПО должно работать с несколькими документами одновременно: с несколькими текстами или изображениями. Однодокументную – если одновременная работа с несколькими документами не обязательна.

Выбор подхода к разработке. Если выбран интерфейс со свободной навигацией или прямого манипулирования, то это предполагает использование событийного программирования и объектного подхода, так как современные среды визуального программирования (Visual C++, Delphi, Builder C++ и др.) предоставляют интерфейсные компоненты именно в виде объектов библиотечных классов. В зависимости от сложности предметной области ПО может реализовываться с использованием объектов и классов и чисто процедурно. Исключение -

случаи использования специализированных языков разработки Интернет-приложений (Perl), построенных по совершенно другому принципу.

Примитивный интерфейс и интерфейс типа меню совместимы со структурным и с объектным подходами к разработке. Поэтому выбор подхода осуществляют с использованием дополнительной информации.

Практика показывает, что объектный подход эффективен для разработки очень больших программных систем (более 100.000 операторов универсального ЯП) и в тех случаях, когда объектная структура предметной области ярко выражена. Объектный подход необходимо осторожно использовать при жестких ограничениях на эффективность разрабатываемого ПО, например, при разработке систем резального времени.

Во всех прочих случаях выбор подхода остается за разработчиком.

Выбор языка программирования. В большинстве случаев проблемы выбора языка программирования реально не существует. Язык может быть определен:

- организацией, ведущей разработку;
- программистом, который по возможности будет использовать знакомый язык;
- устоявшимся мнением и т.п.

Существующие языки программирования можно разделить на группы:

- универсальные языки высокого уровня;
- специализированные языки разработчика программного обеспечения;
- специализированные языки пользователя;
- языки низкого уровня.

Универсальные языки высокого уровня Delphi, C++, C#. Достоинства:

- многоплатформенность;
- наличие операторов, реализующих структурные алгоритмические конструкции;
- возможность программирования на низком уровне с использованием адресов ОП;
- огромные библиотеки подпрограмм и классов.

Специализированные языки используют для создания конкретных типов ПО. Это: языки баз данных; языки создания сетевых приложений; языки создания систем искусственного интеллекта и т.д. Специализированные языки пользователя являются частью профессиональных сред пользователя и характеризуются узкой направленностью.

Языки низкого уровня позволяют осуществлять программирование практически на уровне машинных команд. При этом получают самые оптимальные с точки зрения времени выполнения и с точки зрения объема необходимой памяти программы. Эти языки не годятся для создания больших программ и программных систем. Основная причина – низкий уровень абстракций, которыми должен оперировать разработчик, откуда большое время разработки.

Существенно и то, что сами языки низкого уровня не поддерживают принципов структурного программирования, что значительно ухудшает технологичность разрабатываемых программ.

Языки типа Ассемблера обычно используют:

• при написании сравнительно простых программ, взаимодействующих непосредственно с техническими средствами (драйверов);

• в виде вставок в программы на языках высокого уровня, например, для ускорения преобразования данных в циклах с большим количеством повторений.

Выбор среды программирования. *Среда программирования* называют программный комплекс, который включает специализированный текстовый редактор, встроенные компилятор, компоновщик, отладчик, справочную систему и другие программы, использование которых упрощает процесс написания и отладки программ.

Широко распространены среды визуального программирования, в которых программист получает возможность визуального подключения к программе некоторых кодов из специальных библиотек компонентов, что стало возможным с развитием ООП.

Визуальные среды - Delphi, C++ Builder, Visual C++, Visual Studio и др. Выбор между этими средами должен определяться характером проекта.

Выбор или формирование стандартов разработки. Современная технология проектирования должна обеспечивать соответствие стандарту ISO/IEC 12207: 1995 (поддержка всех процессов ЖЦ ПО).

Реальное применение любой технологии проектирования ПО требует формирования или выбора ряда стандартов (правил, соглашений), которые должны соблюдаться всеми участниками проекта. К таким стандартам относятся:

• стандарт проектирования;

• стандарт оформления проектной документации;

• стандарт интерфейса пользователя.

Стандарт проектирования должен определять:

• набор необходимых моделей (схем, диаграмм) на каждой стадии проектирования и степень их детализации;

• правила фиксации проектных решений на диаграммах, в том числе правила именования объектов и соглашения по терминологии, набор атрибутов для всех объектов и правила их заполнения на каждой стадии, правила оформления диаграмм и т.д.;

• требования к конфигурации рабочих мест разработчиков, включая настройки операционной системы и используемых CASE-средств;

• механизм обеспечения совместной работы над проектом, в том числе и правила интеграции подсистем проекта и анализа проектных решений на непротиворечивость.

Стандарт оформления проектной документации должен регламентировать:

• комплектность, состав и структуру документации на каждой стадии;

• требования к оформлению документации (включая требования к содержанию разделов, подразделов, пунктов, таблиц и т.д.);

• правила подготовки, рассмотрения, согласования и утверждения документации с указанием предельных сроков на каждой стадии;

• требования к настройке CASE – средств для обеспечения подготовки документации в соответствии с установленными правилами.

Стандарт интерфейса пользователя должен определять:

• правила оформления экранов (шрифты и цветовую палитру), состав и расположение окон и элементов управления;

• правила пользования клавиатурой и мышью;

• правила оформления текстов помощи;

• перечень стандартных сообщений;

• правила обработки реакции пользователя.

Все описанные выше проектные решения существенно влияют на трудоемкость и сложность разработки. Только после их принятия следует переходить к анализу требований и разработке спецификаций проектируемого ПО.