МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Кафедра информационных и автоматизированных производственных систем

Составитель
А. В. Матисов

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания к самостоятельной работе

Рекомендовано цикловой методической комиссией специальности
СПО 09.02.07 Информационные системы и программирование
в качестве электронного издания
для использования в образовательном процессе

Кемерово 2018

Рецензенты

Ванеев О. Н. – кандидат технических наук, доцент кафедры информационных и автоматизированных производственных систем

Чичерин И. В. – кандидат технических наук, доцент кафедры информационных и автоматизированных производственных систем

Матисов Александр Вениаминович

Инструментальные средства разработки программного обеспечения: методические указания к самостоятельной работе [Электронный ресурс] для студентов специальности СПО 09.02.07 Информационные системы и программирование очной формы обучения / сост. А. В. Матисов; КузГТУ. – Электрон. издан. – Кемерово, 2018.

Приведено содержание самостоятельной работы, материал, необходимый для успешного изучения дисциплины.

Назначение издания – помощь обучающимся в получении знаний по дисциплине «Инструментальные средства разработки программного обеспечения» и организация самостоятельной работы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью самостоятельной работы обучающихся является получение новых знаний по дисциплине «Инструментальные средства разработки программного обеспечения».

Самостоятельная работа необходима для формирования у обучающихся способности самостоятельно решать задачи профессиональной деятельности, формирования умения и навыков планирования времени, формирования стремления развиваться и совершенствоваться.

Часть 1 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО САМОСТОЯТЕЛЬНОЙ ПОДГОТОВКЕ К ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

1.1 ИСТОРИЯ РАЗВИТИЯ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ РАЗРАБОТКИ ПРОГРАММ (ИСРП)

Программные продукты можно классифицировать по различным признакам. Рассмотрим классификацию, в которой основополагающим признаком является сфера (область) использования программных продуктов:

- аппаратная часть автономных компьютеров и сетей ЭВМ;
- функциональные задачи различных предметных областей;
- технология разработки программ.

В этих областях выделим соответственно три класса программных продуктов (рисунок 1.1):

- системное программное обеспечение;
- пакеты прикладных программ;
- инструментарий технологии программирования.

Рисунок 1.1 – Классификация программных продуктов

Очевидно, что в рамках нашей дисциплины нас интересует третье направление.

Развитие этого направления, связанно с переходом на промышленную технологию производства программ, стремлением к сокращению сроков, трудовых и материальных затрат на производство и эксплуатацию программ, обеспечению гарантированного уровня их качества.

В рамках этого направления сформировались следующие группы программных продуктов:

- средства для создания приложений, включающие:
  - локальные средства, обеспечивающие выполнение отдельных работ по созданию программ;
  - интегрированные среды разработчиков программ, обеспечивающие выполнение комплекса взаимосвязанных работ по созданию программ;
- CASE-технология (Computer-Aided System Engineering), представляющая методы анализа, проектирования и создания программных систем и предназначенная дли автоматизации процессов разработки и реализации информационных систем.

Рассмотрим подробнее эти группы программных продуктов.

Средства для создания приложений – совокупность языков и систем программирования, а также различные программные комплексы для отладки и поддержки создаваемых программ.

Локальные средства разработки программ. Эти средства на рынке программных продуктов наиболее представительны и включают языки и системы программирования, а также инструментальную среду пользователя.

Язык программирования – формализованный язык для описания алгоритма решения задачи на компьютере.

Системы программирования *(programming system)* включают:

- компилятор;
- интегрированную среду разработчика программ;
- отладчик;
- средства оптимизации кода программ;
- набор библиотек (возможно с исходными текстами программ);
- редактор связей;
- сервисные средства (утилиты) для работы с библиотеками, текстовыми и двоичными файлами;
- справочные системы;
- документатор исходного кода программы;
- систему поддержки и управления проектом программного комплекса.

Средства поддержки проектов – новый класс программного обеспечения, предназначен для:

- отслеживания изменений, выполненных разработчиками программ;
- поддержки версий программы с автоматической разноской изменений;
- получения статистики о ходе работ проекта.

Инструментальная среда пользователя представлена специальными средствами, встроенными в пакеты прикладных программ, такими, как:

- библиотека функций, процедур, объектов и методов обработки;
- макрокоманды;
- клавишные макросы;
- языковые макросы;
- программные модули-вставки;
- конструкторы экранных форм и отчетов;
- генераторы приложений;
- языки запросов высокого уровня;
- языки манипулирования данными;
- конструкторы меню и многое другое.

Интегрированные среды разработки программ. Дальнейшим развитием локальных средств разработки программ, которые объединяют набор средств для комплексного применения на всех технологических этапах создания программ, являются интегрированные программные среды разработчиков. Основное назначение инструментария данного вида – повышение производительности труда программистов, автоматизация создания кодов программ, обеспечивающих интерфейс пользователя графического типа, разработка приложений для архитектуры клиент-сервер, запросов и отчетов.

CASE-технология создания информационных систем.

Средства CASE-технологии – относительно новое, сформировавшееся на рубеже 80-х годов направление. CASE-технология – программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем.

В истории развития ИСРП обычно выделяют шесть периодов. Периоды различаются применяемой техникой и методами

разработки программных средств (ПС). В эти периоды, в качестве инструментов разработки, используют следующие ПС:

Период 1. Ассемблеры, анализаторы.

Период 2. Компиляторы, интерпретаторы, трассировщики.

Период 3. Символические отладчики, пакеты программ.

Период 4. Системы анализа и управления исходными текстами.

Период 5. Первое поколение CASE. Это CASE – средства, позволяющие выполнять поддержку начальных работ процесса разработки ПС и систем. Адресованы непосредственно системным аналитикам, проектировщикам, специалистам в предметной области. Поддерживают графические модели, экранные редакторы, словари данных. Не предназначены для поддержки полного жизненного цикла (ЖЦ) ПС.

Период 6. Второе поколение CASE. Представляют собой набор инструментальных средств, каждое из которых предназначено для поддержки отдельных этапов процесса разработки или других процессов ЖЦ. В совокупности обычно поддерживают практически полный ЖЦ. Используют средства моделирования предметной области, графического представления требований, поддержки автоматической кодогенерации. Содержат средства контроля и управления разработкой, интеграции системной информации, оценки качества результатов разработки. Поддерживают моделирование и прототипирование системы, тестирование, верификацию, анализ сгенерированных программ, генерацию документации по проекту.

Наиболее востребованы CASE-средства на первых этапах ЖЦ, связанных с анализом требований и проектированием. CASE – средства позволяют использовать визуальные среды разработки, средства моделирования и быстрого прототипирования разрабатываемой системы. Это позволяет как можно раньше оценить, насколько будущая система устраивает заказчика и насколько она дружественна будущему пользователю.

1.2 ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА УПРАВЛЕНИЯ ПРОЕКТОМ

1.2.1 Обзор возможностей инструментальных средств управления проектом

Управление проектом объединяет множество инструментальных средств, методов и технологий, связанных с планированием проекта и управлением интегрированным процессом.

Управление проектом – нечто вроде распределения и управления бюджетом, временем и людьми. Давно проверенный практикой принцип гласит, что если вы не сможете спланировать что-то, вы не сможете это и сделать. Планирование требует некоторых начальных знаний того, что требуется для разработки системы в этой организации, с этими людьми, с этими ресурсами, в этой организационной культуре и т. п. Организация должна оценить, что происходило с предыдущими проектами, определить из предыдущих проектов метрики (систему показателей).

Современные инструментальные средства управления проектами объединяют обычное управление проектом со стратегическим планированием, бизнес-моделированием, портфельным управлением, управлением документацией, технологическим процессом и т. д. Лучшие представители управления большими проектами используют веб-технологии. Которые позволяют осуществлять динамическое, интерактивное и синхронное управление большими коллективами и процессами.

Одним из главных инструментов управления планированием проектом являются сетевые графики. Два традиционных вида графиков операций – метод критического пути и диаграмма Гантта.

Инструментальные средства управления проектами могут формировать график операции, как только будет обеспечена необходимая проектная информация. Требуемая проектная информация включает определение операций, их продолжительность, ресурсы, предшествующие события и события которые являются следствием операций.

Инструментальные средства управления проектами регулируют и перестраивают планы всякий раз, когда происходят изме-

нения в операциях или ресурсах. Это позволяет менеджерам понять, может ли дополнительный ресурс дать лучший результат в проекте.

1.2.2 Управление проектом в программе MS PROJECT

Несмотря на пользовательский интерфейс, схожий с приложениями Microsoft Office, Microsoft Project является специализированной программой, предназначенной исключительно для управления проектами.

Семейство продуктов Microsoft Project состоит из трех приложений MS Project Standard, MS Project Professional, MS Project Server. MS Project Standard и MS Project Server переведены на русский язык.

Стандартная и профессиональная версия программы предназначены для создания плана проекта, который затем можно опубликовать на сервере MS Project Server для организации коллективной работы над проектом.

С помощью сервера MS Project Server члены проектной команды получают от руководителя задачи, сообщают о ходе их выполнения. Руководитель проекта в соответствии с данными, поступающими от сотрудников, отслеживает ход выполнения работ и анализирует загрузку сотрудников.

В качестве СУБД MS Project Server использует MS SQL Server последних версий.

Для совместной работы над документами и задачами предназначен пакет Microsoft SharePointTeam Services, входящий в состав MS Project Server.

Основные термины.

Задача – активность, направленная на достижение определенного результата.

Ресурс назначается на задачу и может быть материальным (материалы, оборудование и сырье) и рабочим (работники и оборудование, оплачиваемое по часам).

Выделение ресурса на задачу называется назначением. На одну задачу можно назначить неограниченное количество ресурсов.

Задача характеризуется длительностью, объемом трудозатрат (работ) и стоимостью, необходимыми для ее выполнения.

Длительность проекта складывается из промежутков времени от начала самой ранней задачи до окончания самой поздней с учетом зависимостей между задачами.

Планирование проектов.

В деятельности любой организации различают программы, операции и проекты. Программа – это деятельность по управлению проектами и операциями внутри организации. Операции – это повторяющиеся действия, направленные на поддержание деловой активности. Проект же предпринимается для достижения определенного результата в условиях установленных сроков и средств.

Управление проектами заключается в составлении плана и отслеживании выполнения работ по нему. Чем лучше составлен план проекта, тем легче выполнять проектные работы и удачно завершить проект.

Проекты могут быть любого масштаба, в них может быть задействовано несколько человек, а может и несколько тысяч. Проекты могут иметь разной длительности от нескольких дней до нескольких лет. Проекты могут осуществляться в любой области деятельности.

Для каждого проекта четко определены начало и конец. Конец проекта наступает с достижением поставленных целей или когда становится ясно, что цели не могут быть достигнуты и проект обрывается.

После окончания проекта проектная команда распадается, и ее участники переходят в другие проекты.

План проекта составляется, чтобы определить, с помощью каких работ будут достигнуты результаты проекта, какие люди и оборудование нужны для выполнения этих работ, и в течение какого времени люди и оборудование будут заняты работой по проекту. План проекта содержит три основных элемента: задачи, ресурсы и назначения.

Задачей называется работа, осуществляемая в рамках проекта для достижения определенного результата.

Фаза проекта состоит из одной или нескольких задач, в результате выполнения которых достигается один или несколько результатов проекта. Таким образом, результат фазы суммирует результаты входящих в нее задач. Совокупность фаз проекта на-

зывается его жизненным циклом. Фазы могут состоять как из задач, так и из других фаз.

В каждом проекте для достижения общей цели необходимо достигнуть нескольких промежуточных целей. Задачи, в результате выполнения которых достигаются промежуточные цели, называются завершающими задачами. В MS Project они называются вехами. Вехи могут присутствовать в проекте для облегчения отслеживания плана проекта.

Длительность задачи – это период рабочего времени, который необходим для ее выполнения. Длительность может не соответствовать трудозатратам работников, которые выполняют эту задачу.

Задачи в плане проекта взаимосвязаны. Зависимости между задачами обозначаются связями и используются, чтобы задать логику, определяющую последовательность работ в проекте.

Ресурсы в MS Project делятся на трудовые и материальные и обозначают сотрудников и оборудование, необходимые для выполнения проектных задач. При составлении списка ресурсов часто используется ролевое планирование, то есть для выполнения определенной задачи назначается роль, а конкретный сотрудник выбирается на эту роль на этапе реализации проекта.

Для каждого ресурса имеется важное свойство – его стоимость. В MS Project есть два типа стоимости ресурсов: повременная ставка и стоимость за использование. Повременная ставка выражается в стоимости использования ресурса в единицу времени. Обычно повременная ставка используется для учета трудовых ресурсов. Затраты на использование начисляются всякий раз, когда в задаче используется ресурс.

Назначение – это связь определенной задачи и ресурсов, необходимых для ее выполнения. Для выполнения задачи могут быть назначено несколько трудовых и материальных ресурсов.

Три составляющих проекта – время, стоимость, объем работ образуют проектный треугольник. При изменении одной из составляющих меняются две другие. Все три составляющие тесно связаны между собой. Например, если уменьшается время, то возрастет стоимость или уменьшится объем.

При изменении любой из трех составляющих надо убедиться, что это не приведет к уменьшению – запланированного уровня качества проекта.

Планирование проекта в MS Project.

Планирование начинается с определения проекта, то есть описания его ключевых характеристик. Затем составляется список задач и фаз и список необходимых для их выполнения ресурсов. Вносится дополнительная информация о задачах и ресурсах. Осуществляются назначения, после чего проект оптимизируется.

Определение проекта состоит в задании его ключевых характеристик. Эти характеристики задаются при создании нового файла проекта.

Планирование стоимости в MS Project.

Общая стоимость проекта складывается из фиксированной стоимости ресурсов и задач и стоимости назначений, которая определяется ставками ресурса, трудозатратами и стоимостью использования ресурса.

Для каждого ресурса можно определить его стоимость использования в проекте: почасовую ставку или стоимость за использование. Стоимость назначения определяется стоимостью ресурса, умноженной на длительность назначения (по часовой ставке), либо фиксированной стоимостью ресурса.

Стоимость ресурса определяется на вкладке «Затраты» диалогового окна «Сведения о ресурсе».

Стоимость назначений определяется автоматически путем умножения ставки ресурса на трудозатраты и прибавлением к результату затрат на использование ресурса. Изменить стоимость назначения можно, указав другую тарификационную таблицу. Это можно сделать на вкладке «Общие» диалогового окна «Сведения о назначении», где из списка можно выбрать таблицу A, B, C, D, E.

Если в проекте часто применяются тарификационные таблицы, то можно в представлении добавить колонку «Таблица норм затрат» для удобства.

Стоимость задач складывается из суммарной стоимости назначений и фиксированных затрат. Фиксированные затраты – это затраты, не связанные с использованием проектных ресурсов. Например, стоимость авиабилетов.

Для ввода фиксированных затрат используется поле «Фиксированные затраты» в таблице «Затраты» в любом представлении для работы с задачами.

Планируя стоимость проекта, необходимо предусмотреть не только его бюджет, но и определить, как этот бюджет будет расходоваться на протяжении проекта. Расходование бюджета зависит от порядка оплаты работ. Оплачивать работу можно поразному: может использоваться предоплата, оплата по факту завершения или оплата по мере выполнения работ. Причем в одном проекте могут сочетаться несколько способов оплаты.

Способ оплаты можно указать и для ресурсов и для фиксированных затрат на задачу. На вкладке «Затраты» в диалоговом окне «Сведения о ресурсах» способ оплаты выбирается в раскрывающемся списке «Начисление затрат».

Определить порядок оплаты фиксированных затрат можно в колонке «Начисление фиксированных затрат» таблицы «Затраты» в любом представлении для работы с задачами.

По умолчанию способ начисления фиксированных затрат выбирается согласно параметра «Начисление фиксированных затрат по умолчанию» на вкладке «Расчет» диалогового окна «Параметры».

Анализ и оптимизация плана работ.

В ходе анализа плана проекта нужно оценить, насколько установленные длительности задач реалистичны. MS Project имеет средства для вероятностной оценки проекта.

Метод PERT (Program, Evaluation and Review Technique) использует три сценария: пессимистичный (максимальные длительности задач), оптимистичный (минимальные длительности задач) и ожидаемый. В соответствии с удельным весом каждого варианта программа рассчитывает средневзвешенную длительность каждой задачи.

Для анализа проекта по методу PERT надо вывести панель инструментов «Анализ по методу PERT».

Для анализа надо ввести для каждой задачи пессимистичную, оптимистичную ожидаемую длительности. Для этого надо нажать на кнопку «Лист ввода PERT».

Затем надо ввести весовые коэффициенты. Для этого на панели инструментов «Анализ по методу PERT» надо нажать кнопку «Задание весовых коэффициентов для метода PERT».

Сумма весовых коэффициентов всегда равна 6.

После определения весовых коэффициентов можно преступить к расчету для этого надо нажать кнопку «Вычисления по методу PERT». Перед этим следует сохранить файл проекта под другим именем, чтобы можно было вернуться к предыдущим данным.

MS Project выведет окно с предупреждением, что значения полей «Длительность», «Начало» и «Окончание» будут пересчитаны.

Анализ и оптимизация стоимости проекта

При анализе стоимости проекта обычно оценивается его бюджет (суммарные затраты на проект) и соотношение составляющих бюджета. Если общая стоимость проекта превышает ожидания, то бюджет оптимизируется.

Чтобы оценить общую стоимость проекта, достаточно перейти в таблицу «Затраты» в любом представлении со списком затрат и посмотреть данные в столбце «Общие затраты» у суммарной задачи проекта.

Также требуется проанализировать пропорциональное соотношение затрат внутри бюджета. При этом рассматриваются:

- распределение затрат по фазам проекта;
- распределение затрат по типам работ;
- соотношение между затратами на сверхурочные трудозатраты и обычные;
- распределение затрат на ресурсы разных типов.

Для этого можно воспользоваться формулами и настраиваемыми полями.

После проведения анализа может потребоваться провести оптимизацию проекта: уменьшить или увеличить затраты на задачи или ресурсы определенного типа.

Затраты определяются ставками ресурсов, трудозатратами и фиксированными затратами на задачи.

Для уменьшения затрат можно привлечь более дешевые ресурсы или использовать тарификационные таблицы с более низ-

кими ставками. Привлечение более дешевых ресурсов может вызвать снижение качества проекта и увеличение сроков проекта.

Если у проекта оказывается дополнительный бюджет, то можно повысить качество проекта, привлекая более квалифицированные ресурсы и используя более дорогие материалы и оборудование.

Анализ рисков

Анализ опасностей, которые могут возникнуть при выполнении проекта, – один из самых сложных и ответственных этапов подготовки проекта. От качества проведения анализа во многом зависит успешный результат проекта.

Анализ рисков состоит из нескольких этапов:

- определение возможных рисков;
- определить стратегию действий при возникновении подобной ситуации.

Риски могут быть многих типов:

- риски в расписании проекта;
- ресурсные риски;
- бюджетные риски;
- социальные риски;
- политические риски;
- природные.

На обычный проект чаще всего могут оказывать влияние первые три типа рисков.

Для оценки рисков в расписании проекта следует выделить задачи со следующими параметрами: задачи с предварительными длительностями, задачи с большой длительностью и в которых задействовано большое количество ресурсов, задачи с большим числом зависимостей, задачи с внешними зависимостями.

Для оценки ресурсных рисков следует выделить следующие ресурсы: работников с недостаточной квалификацией, ресурсы, чрезмерно загруженные, ресурсы с уникальными навыками, которые трудно заменить.

Для оценки бюджетных рисков требуется определить насколько увеличение объема работы по проекту приведет к увеличению затрат на него и имеются ли бюджетные резервы для компенсации.

Для смягчения рисков требуется разработать стратегию их сдерживания и уменьшения влияния. Для этого надо заложить в проект временной и бюджетный запас, а также предусмотреть меры по дублированию уникальных ресурсов.

Отслеживание проекта в MS Project.

Диаграмма Гантта.

Диаграммы являются графическим средством отображения содержащейся в проектном файле информации. В MS Project диаграммы являются не только средством просмотра проектной информации. С помощью диаграмм можно вводить и редактировать данные.

Диаграмма Гантта – это один из наиболее популярных способов графического представления плана проекта, применяемый во многих программах управления проектами.

Она представляет собой график, на котором по горизонтали расположена шкала времени, а по вертикали – список задач. Длина отрезков, обозначающих задачи пропорциональна длительности задач. Задачи соединяются между собой линиями связей.

Рядом с отрезками может отображаться дополнительная информация, определяемая настройками программы.

В MS Project входят несколько представлений, основанных на диаграмме Гантта: «Диаграмма Гантта», «Подробная диаграмма Гантта», «Диаграмма Гантта с выравниванием», «Диаграмма Гантта с отслеживанием» и т. д.

Форматирование диаграммы Гантта.

Для изменения отображения информации на диаграмме Гантта используется форматирование. С помощью средств форматирования можно изменять форму и цвет составляющих диаграмму фигур, определять информацию, отображаемую рядом с фигурами, отображать дополнительную графическую информацию, форматировать шкалу времени.

Настройка формы и цвета отрезков определяется в диалоговом окне форматирования отрезков. Для открытия окна можно дважды щелкнуть левой кнопкой мыши на отрезке или щелкнуть правой кнопкой мыши и выбрать из выпадающего меню пункт «Форматировать отрезок...».

Диалоговое окно «Шкала времени» имеет четыре закладки. Закладки «Верхний уровень», «Средний уровень», «Нижний уро-

вень» настраивают три уровня временной шкалы. Единицы измерения времени более высокого уровня не могут быть меньше единиц измерения более низкого.

Количество отображаемых уровней временной шкалы задается с помощью списка «Отображать».

Параметры отображения уровня: «Единицы» – определяет единицы измерения; «Интервал» – число единиц в одном делении шкалы уровня; «Надписи» – определяет формат даты.

Закладка «Нерабочее время» задает отображение на диаграмме нерабочих дней. (Воскресных и праздничных).

Для выбора периода времени, отображенного на временной шкале, используется диалоговое окно «Масштаб», которое вызывается из контекстного меню, выбором одноименной команды.

В диалоговом окне можно выбрать период времени, для отображения задач, запланированных на этот период.

Для быстрого изменения масштаба диаграммы Гантта можно воспользоваться кнопками «Увеличить» и «Уменьшить», расположенными на панели инструментов «Стандартная».

Для форматирования вспомогательных линий диаграммы используется диалоговое окно «Сетка». Это окно можно вызвать с помощью команды «Формат» -> «Сетка...» или из контекстного меню.

Для настройки дополнительных параметров диаграммы используется диалоговое окно «Макет». В этом диалоговом окне задаются дополнительные параметры отображения диаграммы: вид линий связи между задачами, формат даты, высоту отрезков и т. д.

Для быстрого форматирования диаграммы Гантта можно использовать мастер диаграмм Гантта. Вызвать мастер можно с помощью команды «Формат» -> «Мастер диаграмм Гантта».

Диалоговое окно «Шкала времени» имеет четыре закладки. Закладки «Верхний уровень», «Средний уровень», «Нижний уровень» настраивают три уровня временной шкалы. Единицы измерения времени более высокого уровня не могут быть меньше единиц измерения более низкого.

Количество отображаемых уровней временной шкалы задается с помощью списка «Отображать».

Параметры отображения уровня: «Единицы» – определяет единицы измерения; «Интервал» – число единиц в одном делении шкалы уровня; «Надписи» – определяет формат даты.

Закладка «Нерабочее время» задает отображение на диаграмме нерабочих дней. (Воскресных и праздничных).

Для выбора периода времени, отображенного на временной шкале, используется диалоговое окно «Масштаб», которое вызывается из контекстного меню, выбором одноименной команды.

В диалоговом окне можно выбрать период времени, для отображения задач, запланированных на этот период.

Для быстрого изменения масштаба диаграммы Гантта можно воспользоваться кнопками «Увеличить» и «Уменьшить», расположенными на панели инструментов «Стандартная».

Для форматирования вспомогательных линий диаграммы используется диалоговое окно «Сетка». Это окно можно вызвать с помощью команды «Формат» -> «Сетка...» или из контекстного меню.

Для настройки дополнительных параметров диаграммы используется диалоговое окно «Макет». В этом диалоговом окне задаются дополнительные параметры отображения диаграммы: вид линий связи между задачами, формат даты, высоту отрезков и т. д.

Для быстрого форматирования диаграммы Гантта можно использовать мастер диаграмм Гантта. Вызвать мастер можно с помощью команды «Формат» -> «Мастер диаграмм Гантта».

1.3 ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОЕКТИРОВАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

1.3.1 Проектирование в среде BPWIN

В структурном подходе к анализу и проектированию используются в основном две группы средств, описывающих функциональную структуру системы и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются:

- DFD (Data Flow Diagrams) – диаграммы потоков данных;
- SADT (Structured Analysis and Design Technique – метод структурного анализа и проектирования) – модели и соответствующие функциональные диаграммы;
- ERD (Entity-Relationship Diagrams) – диаграммы «сущность-связь».

Диаграммы потоков данных и диаграммы «сущность-связь» наиболее часто используемые в CASE-средствах виды моделей.

Конкретный вид перечисленных диаграмм и интерпретация их конструкций зависят от стадии жизненного цикла программного обеспечения (ПО), на которой они применяются.

На стадии формирования требований к ПО SADT-модели и DFD используются, как правило, для моделирования бизнес-процессов (использование SADT-моделей ограничивается только данной стадией, поскольку они не предназначены для проектирования ПО). С помощью ERD выполняется описание данных на концептуальном уровне, не зависимом от средств реализации базы данных (СУБД).

На стадии анализа и проектирования ПО DFD используются для описания структуры проектируемой системы, при этом они могут уточняться, расширяться и дополняться новыми конструкциями. Аналогично ERD уточняются и дополняются новыми конструкциями, описывающими представление данных на логическом уровне, пригодном для последующей генерации схемы базы данных. Вышеперечисленные модели могут дополняться диаграммами, отражающими системную архитектуру ПО, струк-

турные схемы программ, иерархию экранных форм и меню и др. Состав диаграмм в каждом конкретном случае зависит от сложности системы и необходимой полноты ее описания.

Метод SADT разработан Дугласом Россом (SoftTech, Inc.) для моделирования искусственных систем средней сложности.

Данный метод успешно использовался в военных, промышленных и коммерческих организациях США для решения широкого круга задач, таких, как долгосрочное и стратегическое планирование, автоматизированное производство и проектирование, разработка ПО для оборонных систем, управление финансами и материально-техническим снабжением и др. Метод SADT поддерживается Министерством обороны США, которое было инициатором разработки семейства стандартов IDEF (Icam DEFinition), являющегося основной частью программы ICAM (интефированная компьютеризация производства), проводимой по инициативе ВВС США. Метод SADT реализован в одном из стандартов этого семейства – IDEFO, который был утвержден в качестве федерального стандарта США, его подробные спецификации можно найти на сайте http://www.idef.com.

Метод SADT представляет собой совокупность правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями. Основные элементы этого метода основываются на следующих концепциях: графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описывается посредством интерфейсных дуг, выражающих «ограничения», которые, в свою очередь, определяют, когда и каким образом функции выполняются и управляются;

Построение SADT-модели.

Построение SADT-модели заключается в выполнении следующих действий:

- сбор информации об объекте, определение его границ;
- определение цели и точки зрения модели;

- построение, обобщение и декомпозиция диаграмм;
- критическая оценка, рецензирование и комментирование.

Построение диаграмм начинается с представления всей системы в виде простейшего компонента – одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок отражает систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг – они также соответствуют полному набору внешних интерфейсов системы в целом.

Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки определяют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых показана как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом в целях большей детализации.

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, то есть, как уже отмечалось, родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить и из него не может быть ничего удалено.

Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые изображены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах.

Каждая детальная диаграмма является декомпозицией блока из диаграммы предыдущего уровня. На каждом шаге декомпозиции диаграмма предыдущего уровня называется родительской для более детальной диаграммы.

Синтаксис диаграмм определяется следующими правилами:

- диаграммы содержат блоки и дуги;
- блоки представляют функции;
- блоки имеют доминирование (выражающееся в их ступенчатом расположении, причем доминирующий блок располагается в верхнем левом углу диаграммы);

- дуги изображают наборы объектов, передаваемых между блоками;
- дуги изображают взаимосвязи между блоками:
  выход-управление;
  выход-вход;
  обратная связь по управлению;
  обратная связь по входу;
  выход-механизм.

Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма изображают одну и ту же часть системы.

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным.

Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные концы должны соответствовать дугам на исходной диаграмме. Все граничные дуги должны продолжаться на родительской диаграмме, чтобы она была полной и непротиворечивой.

На SADT-диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся (по времени) функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т. д.

Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию.

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая, в свою очередь, может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм.

Для того чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм.

Типы связей между функциями.

Одним из важных моментов при моделировании с помощью метода SADT является точная согласованность типов связей между функциями. Различают по крайней мере связи семи типов (в порядке возрастания их относительной значимости):

- случайная;
- логическая;
- временная;
- процедурная;
- коммуникационная;
- последовательная;
- функциональная.

Случайная связь показывает, что конкретная связь между функциями незначительна или полностью отсутствует. Это относится к ситуации, когда имена данных на SADT-дугах в одной диаграмме имеют слабую связь друг с другом.

Логическая связь – данные и функции собираются вместе благодаря тому, что они попадают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

Временная связь – представляет функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

Процедурная связь – функции сгруппированы вместе благодаря тому, что они выполняются в течение одной и той же части цикла или процесса.

Коммуникационная связь – функции группируются благодаря тому, что они используют одни и те же входные данные и/или производят одни и те же выходные данные.

Последовательная связь – выход одной функции служит входными данными для следующей функции. Связь между элементами на диаграмме является более тесной, чем в рассмотренных выше случаях, поскольку моделируются причинно-следственные зависимости

Функциональная связь – все элементы функции влияют на выполнение одной и только одной функции. Диаграмма, являю-

щаяся чисто функциональной, не содержит чужеродных элементов, относящихся к последовательному или более слабому типу связи.

Одним из способов определения функционально-связанных диаграмм является рассмотрение двух блоков, связанных через управляющие дуги.

1.4 СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Оценка удобства интерфейса, и решений предлагаемых для повышения удобства интерфейса, могут проводиться по следующим правилам.

Правило доступности. Система должна быть настолько понятной. Чтобы пользователь, никогда раньше не видевший ее, но хорошо разбирающийся в предметной области. Мог без всякого обучения ее использовать. Это правило служит некоторым идеалом, к которому можно стремиться, поскольку на практике достичь такой степени доступности почти ни когда не удается.

Правило эффективности. Система не должна препятствовать эффективной работе опытных пользователей, работающих с ней долгое время.

Правило непрерывного развития. Система должна способствовать непрерывному росту знаний, умений и навыков пользователя и приспосабливаться к его меняющемуся опыту. Нарушение непрерывности при переходе от одного набора возможностей к другому также приносит неудобства, поскольку пользователь вынужден разбираться с добавленными возможностями в новом контексте.

Правило соблюдения контекста. Система должна быть согласована с контекстом, в котором ей предстоит работать. В контекст могут входить специфика и объемы входных и выходных данных, тип и цели организации, уровень пользователей, зашумленность помещений и прочее.

Перечисленные правила определяют общие требования, которым должен удовлетворять интерфейс.

Принцип структуризации. Интерфейс пользователя должен быть целесообразно структурирован. Близкие по смыслу, родственные его части должны быть связаны видимым образом, а независимые – разделены; похожие элементы должны выглядеть похоже, а непохожие – различаться.

Принцип простоты. Наиболее распространенные операции должны выполняться максимально просто. При этом должны быть видимые ссылки на более сложные процедуры.

Принцип видимости. Все функции и данные, необходимые для решения определенной задачи. Должны быть видны, когда пользователь пытается ее решить.

Принцип обратной связи. Пользователь должен получать сообщения о действиях системы и о важных событиях внутри нее. Сообщения должны быть информативными, краткими, однозначными и написанными на языке, понятном пользователю.

Принцип толерантности. Интерфейс должен быть гибким и терпимым к ошибкам пользователя. Ущерб от ошибок должен снижаться за счет возможности отмены и повтора действий и за счет разумной интерпретации любых разумных действий пользователя и введенных им данных. По возможности следует избегать обязывающего взаимодействия (модальных диалогов), основанного на ограничении свободы пользователя.

Принцип повторного использования. Следует стараться многократно использовать внутренние и внешние компоненты, обеспечивая тем самым унифицированность интерфейса и сходство между его похожими элементами.

Человекомашинный интерфейс обеспечивает связь между пользователем и компьютером, он позволяет достигать поставленных целей, успешно находить решение поставленной задачи. Взаимодействие – обмен действиями и реакциями на эти действия между компьютером и пользователем. Существует ряд стилей взаимодействия, которые подразделяются на два основных вида:

- использование интерфейса команд – ввод команд текстовыми средствами;
- непосредственное манипулирование.

Количество информации, отображаемой на экране, называется экранной плотностью. Исследования показали, что чем меньше экранная плотность, тем отображаемая информация наиболее доступна и понятна для пользователя, и наоборот, если экранная плотность большая, это может вызвать затруднения в усвоении информации ее ясном понимании. Опытные пользователи могут предпочитать интерфейсы с большой экранной плотностью. Информация на экране может быть сгруппирована и упорядочена в значимые части. Это может быть достигнуто использованием кадров (фреймов), методов типа цветового кодирования,

рамок, негативного изображения и других методов привлечения внимания.

Выделение элементов интерфейса яркостью – это размещение более ярких элементов на фоне более темных. Однако это может вызвать обратный эффект – перегрузку интерфейса.

Существует несколько приемов выделения яркостью:

- движение (мигание или изменение позиции);
- яркость – не очень действенный метод, т. к. люди имеют небольшое число уровней различения яркости;
- цвет;
- форма (символ, шрифт, форма символа;
- использование различных шрифтов в различных формах;
- размер – наиболее эффективно делать различие в 1,5 раза;
- оттенение (различная текстура) объектов;
- окружение (подчеркивание, рамки, инвертированное изображение).

Цвет – мощный визуальный инструмент, и применять его надо очень осторожно, чтобы не вызвать у пользователя дискомфорт от ошибочных цветовых комбинаций.

При разработке эргономичного интерфейса надо учитывать:

- ограничить число цветов до 4 на экране и до 7 – для последовательных экранов;
- для неактивных элементов использовать бледные цвета;
- использовать цвет – как общепринятую кодировку: красный – запрет и т. п.
- учитывать предметное представление о цвете: для картографа: зеленый – лес, желтый – пустыня, синий – вода; для химика: красный – горячо, синий – холодно и т. д.
- для привлечения внимания наиболее эффективны белый, желтый и красный цвета.
- для упорядочения данных используются цвета радуги.
- для разделения данных используются цвета из разных частей спектра: кранный – зеленый; синий – желтый; любой цвет – белый.
- для группировки данных, объединения и подобия нужно использовать соседние цвета спектра: оранжево – желтые, синие - фиолетовые и т. п.

При создании текстовых диалогов необходимо учитывать:

- текст в нижнем регистре читается приблизительно на 13% быстрее, чем текст, полностью напечатанный в верхнем регистре.
- выровненный по правому краю текст читать труднее, чем равномерно распределенный текст с невыровненным правым полем.
- оптимальный интервал между строками равен или немного больше, чем высота символов.

1.5 ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ТЕСТИРОВАНИЯ ПРИЛОЖЕНИЙ

К программным продуктам, рекомендованным к использованию в тестировании на разных этапах относятся:

- Rational Quantify. Профилирование производительности;
- Rational Purify. Отслеживание ошибок с распределением памяти;
- Rational PureCoverage. Отслеживание области покрытия кода;
- Rational TestManager. Планирование тестирования;
- Rational Robot. Выполнение функциональных тестов.

Как при приемочном тестировании, так и при регрессионном разработчик должен предоставить не только работающий код, удовлетворяющий начальным требованиям, но и делающий это максимально безукоризненно с точки зрения производительности и устойчивости. В дополнение, разработчик должен максимально тщательно проверить исполнение всех строчек написанного им кода, во избежание дальнейших несуразностей, ведь одна из часто встречаемых ошибок – это копирование блоков программного кода в разные части программы, при том, что не все ветви первоначального блока были обработаны и протестированы.

Подобные первоначальные ошибки сводят на «нет» дальнейшее функциональное тестирование, так как низкокачественный код не позволяет тестерам быстро писать скрипт, реализующий тестирование новой формы, так как она полна странностей.

Следующие недостатки (относительные) программного кода могут потенциально привести приложение к состоянию зависания (входа в вечный цикл) или к состоянию нестабильности, проявляемой время от времени (в зависимости от внешних факторов):

- Использование рекурсии. Всем известна практическая ценность рекурсии для большого числа задач, особенно математических, но рекурсия является и источником проблем, поскольку не все компиляторы и не во всех случаях способны правильно отрабатывать рекурсию. Посему очень часто в требованиях на

разработку вводится требование на отсутствие рекурсивных функций. Quantify ведет статистический анализ по вызовам функций и она выдает имена всех рекурсивных функций.

– Степень вложенности процедур (функций). В программе можно создать любое число функций (классов), вложить их друг в друга, а потом удивляться почему та или иная функция работает с не той производительностью. Если в требованиях на разработку ограничить полет фантазии разработчиков, то получится более быстрое приложение. Статистику по вложенности и скорости даст Quantify.

– Соглашения о именах. Также оговаривается в требованиях на разработку. Необходимо ввести единый стиль наименования функций, во избежание появления дублей и непонятных имен, что тормозит развитие проекта. Quantify также реализует поиски рассогласовании в именах.

– Использование указателей. Язык С и С++ трудно представить без указателей, даже невозможно. Но, являясь гибким и мощным механизмом обращения к памяти, они же являются минами замедленного действия, проявляющими себя в самые неподходящие моменты. Если проект не может существовать без указателей, то все ошибки, связанные с их неправильной работой позволит отловить Purify.

– Не целевое использование переменных и идентификаторов. Сюда попадают и ошибки связанные с операциями чтения и записи файлов до их открытия или после закрытия. Это и присвоение в переменных до их инициализации. Это и наличие лишних переменных. Эти неточности также могут привести к нестабильности приложения. Purify обрабатывает данную категорию ошибок.

– Не целевое использование блоков данных. Под эту категорию попадают ошибки связанные с распределением памяти, например, невозвращение блоков памяти после их использования. С точки зрения функциональности подобная ошибка не совсем ошибка, так как целостность приложения не нарушается и к сбоям не ведет. Побочный эффект – это замусоривание системы ненужными данными и быстрое «истекание» системных ресурсов. Данный вид отлавливается Purify.

– Присутствие участков кода не исполнявшихся в течении определенного времени. Также потенциальная ошибка, так как не выполненный код может содержать ошибки. PureCoverage отлавливает данный вид ошибок.

– Для осуществления всех функций по тестированию программных модулей, все три продукта используют специальную технологию Object Code Insertion, при которой бинарный файл, находящийся под тестированием, насыщается отладочной бинарной информацией, обеспечивающей сбор информации о ходе тестирования.

– Отметим общие черты для всех трех программных продуктов. Для сбора и обработки информации программам тестирования нужны два файла: исполняемый модуль и его исходный текст. Исполняемый модуль насыщается отладочным кодом, а наличие исходного текста позволяет разработчику легко переключаться между схематическим отображением и кодом тестируемого приложения.

– На код накладываются дополнительные особые ограничения, а именно: тестируемый код должен быть получен при помощи компиляции с опцией «Debug», то есть должен содержать отладочную информацию. В противном случае Quantify, Purify и PureCoverage не смогут правильно отображать (вообще не смогут отображать) имена функций внутренних модулей тестируемого приложения. Исключения могут составлять только вызовы внешних модулей из dll-библиотек, поскольку метод компоновки динамических библиотек позволяет узнавать имена функций.

– Отсюда можно сделать простой вывод: тестировать приложения можно даже не имея отладочной информации в модуле и при отсутствии исходных текстов, но в этом случае разработчик получает статистику исключительно по внешним вызовам.

Программные продукты могут работать в трех режимах:

1. Независимом графическом. В этом случае каждое средство запускается индивидуально а тестирование осуществляется из него в графическом режиме;

2. Независимом командном. Данный режим характеризуется управлением ходом насыщения тестируемого модуля отладочной информации из командной строки;

3. Интегрированном. Этот режим позволяет разработчикам не выходя из привычной среды разработки (Visual Studio .NET) прозрачно вызывать инструменты Quantify, Purify и PureCoverage.

Из дополнительных возможностей всех инструментов хочется отметить наличие специального набора файлов автоматизации, разрешающих разработчикам еще на этапе разработки внедрять С-образные вызовы функций сбора информации по тестированию в свои приложения, получая при этом максимальный контроль над ними.

Средства анализа, встроенные в Quantify, Purify и PureCoverage, позволяют удовлетворить детально контролировать все нюансы в исполнении тестируемого приложения. Здесь и сравнение запусков, и создание слепков в ходе тестирования и экспорт в Excel для построения точных графиков множественных запусков.

Поддерживаются следующие языки программирования:

- Visual C\C++\C# в exe-модулях, dll-библиотеках, ActiveX-компонентах и COM-объектах;
- Поддерживаются проекты на Visual Basic и Java Applets (с любой виртуальной машиной);
- Дополнительно можно тестировать дополнительные модули к MS Word и Ms Excel.

Quantify, вставляя отладочный код в бинарный текст тестируемого модуля, замеряет временные интервалы, которые прошли между предыдущим и текущими запусками. Полученная информация отображается в нескольких видах: табличном, графическом, комбинированном.

Статистическая информация от Quantify позволит узнать какие dll библиотеки участвовали в работе приложения, узнать список всех вызванных функций с их именами, формальными параметрами вызова и с статистическим анализатором, показывающим сколько каждая функция исполнялась.

Гибкая система фильтров Quantify позволяет, не загромождая экран лишними выводами (например, системными вызовами), получать необходимую информацию либо только о внутренних, программных вызовах либо только о внешних, либо комбинируя оба подхода.

Вооружившись полученной статистикой, разработчик без труда выявит узкие места в производительности тестируемого приложения и устранит их в кратчайшие сроки.

Начать описание возможностей продукта Rational Purify хочется перефразированием одного очень известного изречения: «с точностью до миллиБАЙТА». Данное сравнение не случайно, ведь именно этот продукт направлен на разрешение всех проблем, связанных с утечками памяти. Ни для кого не секрет, что многие программные продукты ведут себя «не слишком скромно», замыкая на себя во время работы все системные ресурсы без большой на то необходимости. Подобная ситуация может возникнуть вследствие нежелания программистов доводить созданный код «до ума», но чаще подобное происходит не из за лени, а из-за невнимательности. Это понятно – современные темпы разработки ПО в условиях жесточайшего прессинга со стороны конкурентов не позволяют уделять слишком много времени оптимизации кода, ведь для этого необходимы и высокая квалификация, и наличие достаточного количества ресурсов проектного времени. Как мне видится, имея в своем распоряжении надежный инструмент, который бы сам в процессе работы над проектом указывал на все черные дыры в использовании памяти, разработчики начали бы его повсеместное внедрение, повысив надежность создаваемого ПО. Ведь и здесь не для кого не секрет, что в большинстве сложных проектов первоочередная задача, стоящая перед разработчиками заключается в замещении стандартного оператора «new» в C++, так как он не совсем адекватно себя ведет при распределении памяти.

– Вот на создании\написании собственных велосипедов и позволит сэкономить Purify.

– Общие возможности по управлению Purify схожи с Quantify, за исключением специфики самого продукта. Здесь также можно тестировать многопоточные приложения, также можно делать «слепки» памяти во время тестирования приложения.

– Особенности использования данного приложения касаются спецификой отлавливаемых ошибок и способом выдачи информации.

– Информация выдается в виде списке, с наименованием найденной ошибки или предупреждения. При разворачивании списка с конкретной ошибкой выводится дополнительный набор данных, характеризующих ошибку.

Основное назначение продукта PureCoverage – выявление участков кода, пропущенного при тестировании приложения – проверка области охвата кода.

Очевидно, что при тестировании разработчику или тестировщику не удастся проверить работоспособность абсолютно всех функций. Также невозможно за один проход тестирования исполнить приложение с учетом всех условных ветвлений.

По требованиям на разработку программного кода, программист должен предоставить для функционального тестирования стабильно работающее приложение или модуль, без утечек памяти и полностью протестированный.

Понятие «полностью протестированный» определяет руководство компании в числовом, процентном значении. То есть, при оформлении требований указано, что область охвата кода 70%. Соответственно, по достижении данной цифры дальнейшие проверки кода можно считать нецелесообразными. Конечно, вопрос области охвата, очень сложный и неоднозначный. Единственным утешением может служить то, что 100% области охвата в крупных проектах не бывает.

Из трех рассматриваемых инструментов тестирования PureCoverage можно считать наиболее простым, так как информация им предоставляемая – это просмотр исходного текста приложения, где указано сколько раз исполнилась та или иная строка в приложении.

1.6 ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДОКУМЕНТИРОВАНИЯ ПРИЛОЖЕНИЙ

Документация является органической, составной частью программного продукта для ЭВМ, и требуются значительные ресурсы для ее создания и применения. Тексты и объектный код программ для ЭВМ могут стать программным продуктом только в совокупности с комплексом документов, полностью соответствующих их содержанию и достаточных для его освоения, применения и изменения. Для этого документы должны быть корректными, строго адекватными текстам программ и содержанию баз данных – систематически, структурировано и понятно изложены, для возможности их успешного освоения и использования достаточно квалифицированными специалистами различных рангов и назначения.

Качество и полнота отображения в документах процессов и продуктов в жизненном цикле программных средств должны полностью определять достоверность информации для взаимодействия заказчиков, пользователей и разработчиков, а тем самым корректность функций и достигаемое качество программных продуктов и соответствующих систем. Посредством документов (электронных или бумажных) специалисты взаимодействуют с программными средствами и данными в реализующих их вычислительных машинах, а также между собой.

Существует большая разница между тем, чтобы просто написать и запрограммировать некоторую функцию для индивидуального использования ее разработчиком, и тем, чтобы изготовить ее как качественный программный продукт, отчуждаемый от разработчиков, поставляемый заказчику и пользователям. Создание программного продукта требует значительных организационных усилий, ибо ее документация – это сложный живой организм, подверженный постоянным изменениям, которые могут вноситься многими специалистами. Управление документацией должно непрерывно поддерживать ее полноту, корректность и согласованность с программным продуктом. Необходимо обеспечивать возможность достоверного, формально точного общения всех участников проекта программного средства (ПС) между со-

бой, с создаваемым продуктом и с документами для гарантии поступательного развития, совершенствования и применения комплекса программ. Адекватность документации требованиям, состоянию текстов и объектных кодов программ должна инспектироваться и удостоверяться (подписываться) ответственными руководителями и заказчиками проекта.

Ошибки и дефекты документов не менее опасны для применения ПС, чем ошибки в структуре, интерфейсах, файлах текстов программ и в содержании данных. Поэтому к разработке, полноте, корректности и качеству документации необходимо столь же тщательное отношение, как к разработке и изменениям текстов программ и данных.

Реализация документов ПС в значительной степени определяет достигаемое качество сложных программных продуктов, трудоемкость и длительность их создания. Для этого должны формироваться и использоваться регламентированная стратегия, стандарты, распределение ресурсов и планы создания, изменения и применения документов на программы и данные сложных систем.

В общем случае должны быть выделены руководители и коллектив специалистов, которые будут планировать, описывать, утверждать, выпускать, распространять и сопровождать комплекты документов. Они должны стимулировать разработчиков ПС осуществлять непрерывное, полноценное документирование процессов и результатов своей деятельности, а также контролировать полноту и качество исходных, результирующих и отчетных документов ЖЦ ПС. Официальная, описанная и утвержденная стратегия документирования должна устанавливать дисциплину, необходимую для эффективного создания высококачественных документов на продукты и процессы в жизненном цикле ПС.

Методы и средства документирования каждой процедуры в стандартах обычно не раскрываются и адресуются к специальным нормативным документам различного уровня. Быстро оснащающиеся различными методами и инструментальными средствами этапы системного анализа, моделирования и проектирования ПС различных классов и назначения затрудняют стандартизацию этих процессов, достаточную для полной формализации структуры и содержания документов на программы и данные на уровне международных стандартов. Поэтому для этих этапов

создаются нормативные документы – шаблоны на уровне стандартов де-факто, использующие, адаптирующие и дополняющие компоненты стандартов де-юре в разумной степени. Такие нормативные документы содержат выделенные фрагменты стандартов ЖЦ ПС и других стандартов, регламентирующих шаблоны программных документов на различных этапах проекта. В результате создаются и применяются проблемно-ориентированные совокупности методических руководств, отражающие наиболее современные методы, формы и фрагменты документов для документальной поддержки этапов и процессов жизненного цикла ПС, определенного класса или функционального назначения.

При создании и применении сложных ПС и обеспечении их жизненного цикла документами целесообразно применять выборку из совокупности стандартов, детализирующих частные процессы, работы и документы. В результате на начальном этапе проектирования следует выделять и формировать целесообразный комплект шаблонов документов, обеспечивающих регламентирование всех этапов, процессов и документов при создании определенных проблемно-ориентированных проектов ПС. Для создания и реализации положений этих документов должны быть выбраны инструментальные средства, совместно образующие взаимосвязанный комплекс технологической поддержки и автоматизации ЖЦ и не противоречащие предварительно скомпонованному набору нормативных документов.

Процессы документирования программ и данных входят в весь жизненный цикл сложных систем и ПС. Поэтому организация и реализация работ по созданию документов должны распределяться между специалистами, ведущими непосредственное и преимущественное создание проектов комплексов программ, и специалистами, осуществляющими в основном разработку, контроль и издание документов.

При создании особо сложных систем целесообразно выделение специального коллектива, обеспечивающего организацию и реализацию основных системных работ по документообороту ПС. Совокупные затраты на документирование крупных программных продуктов могут достигать 20–30% от общей трудоемкости проекта и необходимого числа (десятки) специалистов в жизненном цикле проекта ПС. В более простых случаях органи-

зация работ может быть упрощена, затраты на документирование снижаются приблизительно до 10%, однако всегда целесообразно выделять специалистов, непосредственно ответственных за создание, адекватность и контроль полноценного комплекта документов на программный продукт. Состав и общий объем документов широко варьируются в зависимости от класса и характеристик объекта разработки, а также в зависимости от используемой технологии.

Наиболее сложному случаю разработки критических ПС реального времени высокого качества соответствует самая широкая номенклатура документов. Такой перечень документов может быть использован как базовый для формирования на его основе состава и шаблонов документов в остальных более простых проектах.

Создание и применение ПС сложных систем сопровождается документированием этих объектов и процессов их жизненного цикла для обеспечения возможности освоения и развития функций программных средств и баз данных на любых этапах проекта ПС, а также для обеспечения интерфейса между разработчиками и с пользователями. По своему назначению и ориентации на определенные задачи и группы пользователей документацию ПС можно разделить на:

- технологическую документацию процессов разработки и обеспечения всего жизненного цикла, включающую подробные технические описания и подготавливаемую для специалистов, ведущих проектирование, разработку и сопровождение комплексов программ, обеспечивающую возможность отчуждения, детального освоения, развития и корректировки ими программ и данных на всем жизненном цикле ПС;
- эксплуатационную документацию программного продукта – объекта и результатов разработки, создаваемую для конечных пользователей ПС и позволяющую им осваивать и квалифицированно применять эти средства для решения конкретных функциональных задач систем.

Технологическая документация непосредственно и в наибольшей степени должна отражать процессы жизненного цикла комплексов программ и данных и требования к этим документам. Стандарты и нормативные документы, входящие в жизненный цикл проекта ПС, должны регламентировать структуру, состав

этапов, работ и документов ЖЦ ПС. Они должны: формализовать выполнение и документирование конкретных работ при проектировании, разработке и сопровождении ПС; обеспечивать адаптацию документов к характеристикам среды разработки, внешней и операционной системы; регламентировать процессы обеспечения качества ПС и его компонентов, методы и средства их достижения, реальные значения достигнутых показателей качества.

Для контроля возможных изменений целесообразно предусматривать и согласовывать с заказчиком специальный документ, регламентирующий правила применения и корректировки номенклатуры, а также состава и содержания документации, поддерживающей ЖЦ ПС.

Эксплуатационная документация должна обеспечивать отчуждаемость программного продукта от первичных поставщиков – разработчиков и возможность освоения и эффективного применения комплексов программ достаточно квалифицированными специалистами – пользователями.

Эксплуатационные документы должны исключать возможность некорректного использования ПС за пределами условий эксплуатации, при которых документами гарантируются требуемые показатели качества функционирования ПС. Основная ее задача состоит в фиксировании, полноценном использовании и обобщении результатов функционирования объектов и процессов всего жизненного цикла ПС и системы.

Базой эффективного управления проектом ПС и его документированием должен быть План, в котором задачи исполнителей частных работ согласованы с выделяемыми для них ресурсами, а также между собой по результатам и срокам их достижения.

План проекта должен отражать рациональное сочетание целей, стратегий действий, конкретных процедур, доступных ресурсов и других компонентов, необходимых для достижения основной цели с заданным качеством. Планирование реализации проектов и их документирования должно обеспечивать компромисс между характеристиками создаваемой системы и ресурсами, необходимыми на ее разработку и применение.

Реализация плана зависит от результатов выполнения частных работ и документов и может требовать оперативной корректировки плана. Контроль обеспечивает исходные данные для ко-

ординации элементов данной организации в соответствии с планом конкретной задачи. Для этого необходимо следить за ходом проекта и документирования на всем протяжении жизненного цикла и сравнивать запланированные и фактические результаты работ и документы. Контроль является органической функцией управления и должен иметь ряд средств регулирования поведения отдельных специалистов и коллектива разработчиков документов в целом.

При подготовке этих планов целесообразно по возможности разделять их цели и функции. План управления разработкой следует ориентировать на организацию специалистов, непосредственно создающих компоненты и ПС в целом, на эффективное распределение и использование ими ресурсов и средств автоматизации. В Плане управления документированием и обеспечением качества ПС внимание специалистов должно акцентироваться на анализе достигнутых результатов разработки, методах и средствах достижения заданных заказчиком характеристик ПС. При планировании и разработке комплекс документации должен проверяться и проходить аттестацию на полноту в условиях ограниченных ресурсов, на корректность, адекватность и непротиворечивость отдельных документов.

Различия в организационных службах и процедурах, методах и стратегиях приобретения ПС, масштабах и сложности проектов, требованиях систем и методах их разработки влияют на способы разработки, применения и сопровождения документов. Во многих предприятиях используются собственные нормативные шаблоны документов на процессы и продукты ЖЦ ПС, адаптированные к конкретным условиям разработки и характеристикам создаваемых ПС. В них выделяются состав и шаблоны документов, наиболее важные для проекта и качества создаваемого ПС, а также для конкретных заказчиков и пользователей. Соответственно определяются технологические и эксплуатационные документы, для которых регламентируются структура и основное содержание шаблонов документов.

Одна из важнейших задач документирования состоит в том, чтобы увязать четкими экономическими категориями взаимодействие разных специалистов в типовой производственной цепочке: заказчик – разработчик – изготовитель – пользователь докумен-

тации. Для этого объект потребления – программный продукт, его документация и все процессы взаимодействия в цепочке должны быть связаны системой экономических и технических характеристик, в той или иной степени использующих основные экономические показатели – реальные затраты ресурсов: финансов, труда и времени специалистов на конечный программный продукт и документы. Сложность документирования, количество и полнота содержания комплекса документов в первую очередь зависят от масштаба – размера проекта ПС, что целесообразно оценивать в начале его ЖЦ. Для решения этой задачи необходимо детально учитывать требуемые ресурсы современных процессов создания, документирования и использования программ различных классов и назначения – встроенных, коммерческих, административных, учебных, уникальных.

Особое внимание в последнее время уделяется совершенствованию и детализации документов, обеспечивающих высокое качество создаваемых ПС, а также возможности их эффективного итерационного развития длительное время в многочисленных версиях. Соответственно должны изменяться документы, отражающие состояние процессов и компонентов проектов. Для этого организация процессов документирования должна обеспечивать гибкое и точное изменение документов – сопровождение и конфигурационное управление версиями и редакциями каждого документа.

Эти процессы и поддерживающие их средства автоматизации должны быть адекватными тем, которые применяются при сопровождении непосредственных объектов разработки – комплекса программ и данных. Они должны быть поддержаны организацией контроля, регистрации и утверждения версии каждого документа, в той степени и на том уровне, которые необходимы в данном проекте для определенного документа.

Для хранения, тиражирования и распространения документов, сложных ПС высокого качества следует выделять группу специалистов, ответственных за контроль, обеспечение и гарантированное сохранение документации.

Для критических, важных систем документация на программы и данные должна храниться и дублироваться на различных типах носителей и эпизодически выводиться на бумажные

носители. При определении схемы обеспечения сохранности документации разного содержания, следует учитывать ее важность, трудоемкость хранения и возможность аварийного восстановления. Кроме того, должна быть организована служба нормативного контроля, ответственная за соблюдение стандартов, нормативных и руководящих документов при подготовке документации всеми специалистами, участвующими в крупном проекте. Эта служба обязана обеспечить унификацию и высокое качество содержания, структуры и оформления шаблонов документов.

Для обеспечения достоверных данных об объектах и процессах управления документами ПС необходима автоматизированная база данных – информационная система обеспечения и хранения документов проекта. Такая информационная система документооборота представляет собой комплекс формальных и неформальных каналов поэтапного обмена информацией и документами между участниками проекта.

Степень формализации документооборота в коллективе специалистов может варьироваться от утверждаемых руководителями планов, технических заданий и документов на компоненты и версии ПС до личных бесед между разработчиками.

Часть 2 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ДОМАШНИХ ЗАДАНИЙ

Домашние задания по курсу «Инструментальные средства разработки программного обеспечения» включают в себя подготовку к практическим и лабораторным занятиям.

Проработка учебников, изучение конспекта лекций, ознакомление с методическими указаниями по выполнению практических и лабораторных работ позволит студенту всесторонне изучить разбираемую на занятиях тему. Это поможет студенту не только качественно выполнить практические работы, но и успешно пройти текущую и промежуточную аттестации.

СПИСОК ЛИТЕРАТУРЫ

Основная литература

1. Рудаков, А. В. Технология разработки программных продуктов [Электронный ресурс] : учебник для студентов учреждений среднего профессионального образования, обучающихся по специальности «Программное обеспечение вычислительной техники и автоматизированных систем» : [профессиональный модуль ПМ.03 «Участие в интеграции программных модулей» (МДК.03.01)] / А. В. Рудаков. – Москва : Академия, 2017. – 208 с. Режим доступа: http://www.academia-moscow.ru/catalogue/4831/362819/.

Дополнительная литература

1. Вичугова, А. А. Инструментальные средства информационных систем [Электронный ресурс]. – Томск : Издательство Томского политехнического университета, 2015. – 136 с. – Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=442814.

2. Вылегжанина, А. О. Информационно-технологическое и программное обеспечение управления проектом [Электронный ресурс]. – Москва, Берлин : Директ-Медиа, 2015. – 429 с. – Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=362892.

3. Смирнов, А. А. Прикладное программное обеспечение [Электронный ресурс]. – Москва, Берлин : Директ-Медиа, 2017. – 358 с. – Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=457616.

4. Исаченко, О. В. Программное обеспечение компьютерных сетей. – Москва : НИЦ ИНФРА-М, 2017. – 117 с. – Режим доступа: http://znanium.com/go.php?id=851518.