Лабораторная работа №3

Тема: Техническое задание на создание программного продукта.

Цель: Освоение методики предварительного анализа разрабатываемой программы; освоение задач формулирования функциональных и нефункциональных требований к программной реализации отдельных задач и к программе в целом; выработка навыков разработки технического задания.

Задание

1. Установить назначение и общую цель создания программы.
2. Определить структуру программы и состав функциональных задач.
3. Разработать функциональные требования к программе:
   - требования к входным и выходным данным;
   - требования к программной реализации задач;
   - специальные требования к математическому обеспечению программной реализации задач;
   - требованиях к прикладному программному обеспечению.
4. Установить нефункциональные требования к программе:
   - требования к надежности;
   - требования к эффективности;
   - требования к безопасности;
   - требования к эргономичности и удобству использования;
   - требования к численности и квалификации персонала и режиму его работы;
   - требования к переносимости;
   - требования к сопровождаемости;
   - требования к особенностям поставки;
   - требования к защите информации от несанкционированного доступа;
   - требования по сохранению информации при авариях;
   - требования к соответствию стандартам качества.

Теоретические положения

3.1. Объектно-ориентированный подход к моделированию бизнес-процессов

Методология объектно-ориентированного подхода (ООП) тесно связана с концепцией автоматизированной разработки программного обеспечения (CASE – computer aided software engineering). Попытки предложить подходящий синтаксис для визуального представления схем БД и ПО воспринимались разработчиками скептически и неоднозначно. На этом фоне разработка и стандартизация унифицированного языка моделирования UML (unified modeling language) вызвала воодушевление у всего сообщества программистов.

Визуальное моделирование с использованием нотации UML можно представить как процесс поуровневого спуска от наиболее общей и абстрактной

концептуальной модели исходной бизнес-системы к логической модели ИС, а затем и к физической модели соответствующей программного приложения. К базовым моделям UML относятся модель прецедентов (требований) и модель классов (объектную модель).

3.2. Модель вариантов использования

Модель прецедентов (модель требований, модель *вариантов использования*) – это диаграмма UML, на которой изображаются отношения между актерами и вариантами использования. Это исходное концептуальное представление или концептуальная модель системы в процессе ее проектирования и разработки. Создание диаграммы вариантов использования имеет следующие цели:

- определение общих границ и контекста моделируемой предметной области на начальных этапах проектирования ИС;
- формальное представление требований к функциональному поведению (функциональных требований) проектируемой ИС;
- разработка исходной концептуальной модели системы для ее последующей детализации в форме логических и физических моделей в нотации UML;
- модельное обеспечение командной разработки ИС при помощи специальных программных средств управления жизненным циклом ИС;
- подготовка исходной документации для взаимодействия разработчиков системы с ее заказчиками и пользователями.

В общем случае, диаграмма вариантов использования представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров и отношений между этими элементами.

Назначение данной диаграммы состоит в следующем: проектируемая система представляется в форме так называемых вариантов использования (прецедентов), с которыми взаимодействуют внешние сущности или актеры (в английской транскрипции – экторы). При этом актером или действующим лицом называется любой объект, субъект, система или устройство, взаимодействующие с моделируемой бизнес-системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик. Вариант использования служит для описания сервисов, которые система предоставляет актеру в соответствии с его потребностями. При этом не оговаривается, каким образом будет реализовано взаимодействие актеров с системой и собственно выполнение вариантов использования.

Рассматривая диаграмму вариантов использования в качестве модели бизнес-системы, можно ассоциировать ее с "черным ящиком" (см. рис. 2). Концептуальный характер этой диаграммы проявляется в том, что подробная детализация диаграммы или включение в нее элементов физического уровня представления на начальном этапе проектирования скорее имеет

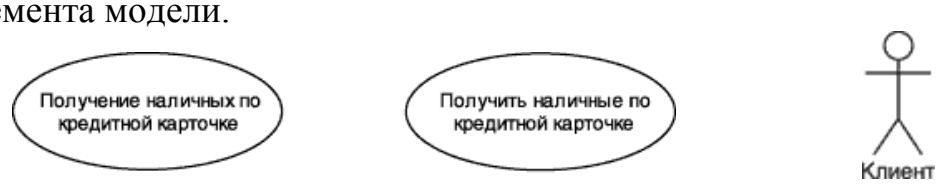
отрицательный характер, поскольку предопределяет способы реализации поведения системы – эти аспекты должны быть сознательно скрыты от разработчика на модели прецедентов.

Базовыми элементами данной диаграммы являются вариант использования и актер.

Вариант использования (use case) – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами, представляет собой спецификацию общих особенностей поведения или функционирования моделируемой системы без рассмотрения ее внутренней структуры.

Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое имя в форме глагольного существительного (рис. 3, а) или глагола (рис. 3, б) с пояснительными словами. Сам текст имени прецедента должен начинаться с заглавной буквы.

Имя (name) – строка текста, которая используется для идентификации любого элемента модели.



а) глагольное существительное б) глагол в) актер

Рис. 3. Графическое обозначение варианта использования и актера

Диаграмма вариантов использования содержит конечное множество вариантов использования, которые в целом должны определять все возможные аспекты ожидаемого поведения системы. Данной модели на всех этапах работы над проектом позволяет не только достичь требуемого уровня унификации обозначений для представления функциональности программы, но и является мощным средством последовательного уточнения требований на основе их итеративного обсуждения со всеми заинтересованными специалистами.

Примерами вариантов использования могут быть следующие действия: занесение систему данных о клиенте, проверка состояния текущего счета клиента, оформление заказа на покупку товара, получение информации о кредитоспособности клиента, отображение графической формы на экране монитора, авторизация пользователя, и другие действия, процессы, задачи.

Актер (actor) – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними, представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Каждый актер может рассматриваться как определенная роль относительно конкретного варианта использования.

Стандартным графическим обозначением актера на диаграммах является фигурка человечка, под которой записывается имя актера (рис. 3, в).

Имя актера должно начинаться с заглавной буквы и следовать рекомендациям использования имен для типов и классов модели, должно быть достаточно информативным с точки зрения семантики. Это может быть наименование должности (например, продавец, кассир, менеджер, и т.д.). Не рекомендуется давать актерам имена собственные или названия моделей конкретных устройств, даже если это с очевидностью следует из контекста проекта, так как одно и то же лицо (устройство) может выступать в нескольких ролях и, соответственно, обращаться к различным сервисам проектируемой системы. При этом в каждый момент времени с системой взаимодействует вполне определенный актер, который играет или выступает в одной из таких ролей. Актеры используются для моделирования внешних по отношению к системе сущностей, которые взаимодействуют с системой. Это могут быть пользователи системы, технические устройства (например, датчики, терминалы обслуживания и т.д.), другие системы. В отдельных случаях в качестве актеров могут выступать подсистемы проектируемой системы, либо ее отдельные классы.

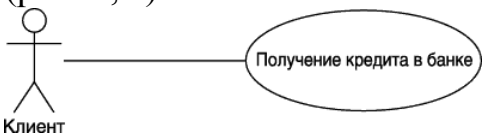
3.3. Отношения на диаграмме вариантов использования.

Отношение (*relationship*) – семантическая связь между отдельными элементами модели. Между элементами модели прецедентов могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов использования. Один актер может взаимодействовать с несколькими вариантами использования. Это значит, что этот актер обращается к нескольким сервисам данной системы. В свою очередь один прецедент может взаимодействовать с несколькими актерами, предоставляя свой сервис для всех них. В то же время два варианта использования, определенные в рамках одной моделируемой системы, могут взаимодействовать друг с другом, однако характер этого взаимодействия будет отличаться от взаимодействия с актерами.

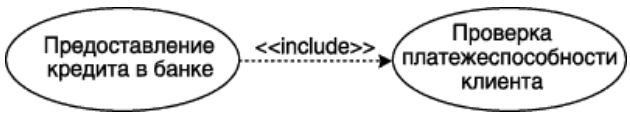
В обоих случаях способы взаимодействия элементов модели предполагают обмен сигналами или сообщениями, которые инициируют реализацию функционального поведения моделируемой системы. В языке UML имеется несколько стандартных видов отношений между элементами модели прецедентов: ассоциация, включение, расширение и обобщение.

С помощью отношений включения, расширения и обобщения могут быть представлены общие свойства вариантов использования. Отношение же ассоциации – одно из фундаментальных понятий в языке UML и в той или иной степени используется при построении всех графических моделей систем в форме канонических диаграмм. На диаграмме прецедентов устанавливается только между актером и вариантом использования и служит для обозначения специфической роли актера при его взаимодействии с прецедентом. На диаграмме вариантов использования, так же как и на других диаграммах, ассоциация обозначается сплошной линией между актером и прецедентом,

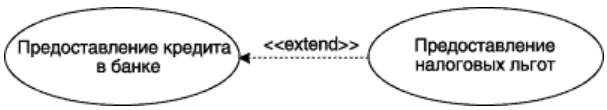
которая может иметь дополнительные обозначения, например, имя и кратность (рис. 4, а).



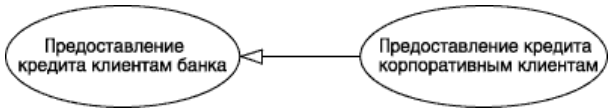
а) отношение ассоциации



б) отношение включения



в) отношение расширения



г) отношение обобщения

Рис. 4. Пример графического представления отношений на диаграмме вариантом использования

В контексте модели прецедентов ассоциация между актером и вариантом использования может указывать на то, что актер инициирует соответствующий прецедент. Такого актера называют главным. В других случаях подобная ассоциация может указывать на актера, которому предоставляется справочная информация о результатах функционирования моделируемой системы. Такого актера называют второстепенным.

Включение (include) в языке UML – это разновидность отношения зависимости между базовым вариантом использования и его специальным случаем. При этом отношением зависимости (dependency) является такое отношение между двумя элементами модели, при котором изменение одного элемента (независимого) неизбежно приводит к изменению другого элемента (зависимого).

Отношение включения устанавливается только между двумя вариантами использования и указывает на то, что заданное поведение для одного варианта использования включается в качестве обязательного составного фрагмента в последовательность поведения другого варианта использования. Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров вариантов использования всегда упорядочена в этом отношении.

Так, например, отношение включения, направленное от варианта использования "Предоставление кредита в банке" к варианту использования "Проверка платежеспособности клиента", указывает на то, что каждый экземпляр первого (базового) прецедента всегда включает в себя функциональное поведение или выполнение второго (включаемого) прецедента, то есть поведение второго прецедента является частью поведения первого прецедента на данной диаграмме. Графически данное отношение обозначается как отношение зависимости в форме пунктирной линии со стрелкой, направленной от базового прецедента к включаемому, при этом данная линия помечается стереотипом <<include>>, как показано на рис. 4, б. Семантика этого отношения определяется следующим образом. Процесс выполнения базового варианта использования включает в себя как собственное

подмножество последовательность действий, которая определена для включаемого варианта использования. Причем выполнение включаемой последовательности действий происходит всегда при инициировании базового прецедента.

Один вариант использования может входить в несколько других прецедентов, а также включать в себя другие прецеденты. Включаемый вариант использования является независимым от базового прецедента в том смысле, что он предоставляет последнему инкапсулированное поведение, детали реализации которого скрыты от последнего и могут быть легко перераспределены между несколькими включаемыми прецедентами. Более того, базовый вариант использования зависит только от результатов выполнения включаемого в него варианта использования, но не от структуры включаемых в него прецедентов.

Расширение (extend) определяет взаимосвязь базового прецедента с другим прецедентом, функциональное поведение которого задействуется базовым не всегда, а только при выполнении дополнительных условий.

В языке UML отношение расширения устанавливается только между вариантами использования, и является зависимостью, направленной к базовому прецеденту и соединенной с ним в так называемой точке расширения. Отношение расширения между вариантами использования обозначается как отношение зависимости в форме пунктирной линии со стрелкой, направленной от прецедента, который является расширением для базового прецедента, и помеченная стереотипом <<extend>>, как показано на рис. 4, в. В изображенном фрагменте имеет место отношение расширения между базовым прецедентом "Предоставление кредита в банке" и прецедентом "Предоставление налоговых льгот". Это означает, что свойства поведения первого прецедента в некоторых случаях могут быть дополнены функциональностью второго прецедента, для чего должно быть выполнено определенное логическое условие данного отношения расширения.

Отношение расширения позволяет моделировать поведение системы таким образом, что один из вариантов использования должен присоединять к своему поведению последовательность действий, определенную для зависимого (расширяющего) прецедента. При этом, в отличие от отношения включения, данное отношение всегда предполагает проверку условия и ссылку на точку расширения в базовом прецеденте. Точка расширения определяет место в базовом прецеденте, в которое должно быть помещено расширение при выполнении соответствующего логического условия. Один вариант использования может быть расширением для нескольких базовых прецедентов, а также иметь в качестве собственных расширений другие прецеденты. Любой базовый вариант использования не зависит от своих расширений.

Семантика отношения расширения определяется следующим образом. Если базовый прецедент выполняет некоторую последовательность действий, которая определяет его поведение, и при этом имеется точка расширения на экземпляр другого прецедента, которая является первой из всех точек расширения у базового прецедента, то проверяется логическое условие данного

отношения. Если это условие выполняется, исходная последовательность действий расширяется посредством включения действий расширяющего варианта использования. Следует заметить, что условие отношения расширения проверяется лишь один раз – при первой ссылке на точку расширения, и если оно выполняется, то все зависимые прецеденты вставляются в базовый прецедент.

Обобщение представляет собой реализацию принципа наследования в ООП. Помимо вариантов использования, может также связывать два и более актера, которые могут иметь общие свойства, т.е. взаимодействовать с одним и тем же множеством вариантов использования одинаковым образом. Такая общность свойств и поведения представляется в виде отношения обобщения с другим, возможно, абстрактным прецедентом/актером, который моделирует соответствующую общность действий/ролей.

Графически отношение обобщения обозначается сплошной линией со стрелкой в форме треугольника без заливки, которая указывает на родительский прецедент или действующее лицо (рис. 4, г). Эта линия со стрелкой имеет специальное название – стрелка-обобщение, – которая используется также и в других канонических диаграммах UML. В данном примере отношение обобщения указывает на то, что вариант использования "Предоставление кредита корпоративным клиентам" есть специальный случай варианта использования "Предоставление кредита клиентам банка". Другими словами, первый прецедент является специализацией второго прецедента. При этом вариант использования "Предоставление кредита клиентам банка" называют предком или родителем по отношению к варианту использования "Предоставление кредита корпоративным клиентам", а последний прецедент называют потомком по отношению к первому прецеденту. Следует отметить, что потомок наследует все свойства поведения своего родителя, а также может обладать дополнительными особенностями поведения.

Отношение обобщения между элементами модели прецедентов применяется в том случае, когда необходимо отметить, что дочерние элементы обладают всеми особенностями поведения родительских. Дочерние актеры/прецеденты участвуют во всех отношениях родительских актеров/прецедентов. По аналогии с принципом наследования ООП, дочерние актеры/прецеденты могут наделяться новыми свойствами поведения, которые отсутствуют у их предков, а также уточнять или модифицировать наследуемые от них свойства поведения.

3.4. Модель классов

Диаграмма классов (class diagram) – диаграмма языка UML, на которой представлена совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения. Предназначена для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы, т.е. графическое

представление таких структурных взаимосвязей логической модели системы, которые не зависят от времени.

Класс (class) – абстрактное описание множества однородных объектов, имеющих одинаковые атрибуты, операции и отношения с объектами других классов.

Графически класс в нотации UML изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции (рис. 5). В этих секциях указываются имя класса, атрибуты и операции класса.

На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником, в котором должно быть указано его имя (рис. 5, а). По мере проработки отдельных компонентов диаграммы описания классов дополняются атрибутами (рис. 5, б) и операциями (рис. 5, в). Четвертая секция (рис. 5, г) не обязательна и служит для размещения дополнительной информации справочного характера, например, об исключениях или ограничениях класса, сведения о разработчике или языке реализации класса. Предполагается, что окончательный вариант диаграммы содержит наиболее полные описания классов, которые состоят из трех или четырех секций.

Рис. 5. Варианты графического изображения класса на диаграмме

Имя класса должно быть уникальным в пределах пакета, который может содержать одну или несколько диаграмм классов. Имя указывается в самой верхней секции прямоугольника, поэтому она называется секцией имени класса. В дополнение к общему правилу именования элементов языка UML, имя класса записывается по центру секции полужирным шрифтом и должно начинаться с заглавной буквы. Рекомендуется в качестве имен классов использовать имена существительные, записанные по практическим соображениям без пробелов. Необходимо помнить, что имена классов образуют словарь предметной области при проектировании системы.

В секции имени класса могут также находиться стереотипы или ссылки на стандартные шаблоны, от которых образован данный класс и, соответственно, от которых он наследует свои свойства: атрибуты и операции.

В этой секции может также приводиться информация о разработчике класса и статус состояния разработки.

Атрибут (attribute) – содержательная характеристика класса, описывающая множество значений, которые могут принимать отдельные объекты этого класса. Атрибут класса служит для представления отдельного свойства или признака, который является общим для всех объектов данного класса. Атрибуты класса записываются в секции атрибутов (рис. 5, б).

Операция (operation) – это сервис, предоставляемый каждым экземпляром или объектом класса по требованию своих клиентов, в качестве которых могут выступать другие объекты, в том числе и экземпляры данного класса. Операции класса записываются в секции операций (рис. 5, в). Совокупность операций характеризует функциональный аспект поведения всех объектов данного класса, который описывается другими каноническими диаграммами UML. Имя операции представляет собой строку текста, которая используется в качестве идентификатора соответствующей операции и должна быть уникальной в пределах данного класса. Имя операции должно начинаться со строчной буквы, и, как правило, записываться без пробелов.

Идентификация классов анализа заключается в предварительном определении набора классов системы (так называемых, классов анализа) на основе описания предметной области и спецификации требований к системе. Прежде всего, выделяются основные абстракции предметной области. Способы идентификации основных абстракций аналогичны способам идентификации сущностей в модели "сущность-связь" – поиск абстракций, описывающих физические или материальные объекты, процессы и события, роли людей, организации и другие понятия предметной области. Единственным формальным способом идентификации сущностей является анализ текстовых описаний предметной области, выделение из описаний имен существительных и выбор их в качестве "кандидатов" на роль абстракций. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Далее абстракции идентифицируются по трем типам, которые используются для уточнения семантики отдельных классов при построении различных диаграмм:

1. Управляющий класс (control class) – класс, отвечающий за координацию действий других классов на диаграмме. На каждой диаграмме классов должен быть хотя бы один управляющий класс, который контролирует последовательность выполнения действий данного варианта использования. Данный класс является активным и инициирует рассылку множества сообщений другим классам модели. Управляющий класс изображается в форме прямоугольника класса со стереотипом <<control>> в секции имени. Примеры управляющих классов: менеджер транзакций, журнал операций, координатор ресурсов, обработчик событий, обработчик ошибок, и т.п.
2. Класс-сущность (entity class) – пассивный класс, информация о котором должна храниться постоянно и не уничтожаться с выключением системы или завершением работы программы. Как правило, этот класс

соответствует отдельной сущности логической модели данных системы. В этом случае его атрибуты являются потенциальными полями таблицы, а операции – присоединенными или хранимыми процедурами. Этот класс пассивный и лишь принимает сообщения от других классов модели, реагируя на них посредством своих операций.

Источники выявления классов-сущностей:

- ключевые абстракции, созданные в процессе архитектурного анализа,
- глоссарий проекта,
- описание потоков событий вариантов использования.

Класс-сущность может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<entity>> с секции имени класса.

3. Граничный класс (boundary class) – класс, который располагается на границе системы с внешней средой и непосредственно взаимодействует с актерами, но, тем не менее, является составной частью системы.

Как правило, для каждой пары "действующее лицо – вариант использования" определяется один граничный класс. Типы граничных классов:

- пользовательский интерфейс, выражающий обмен информацией с пользователем без деталей интерфейса: кнопок, списков, окон, и т.п.
- системный интерфейс с надсистемой или другой системой (используемые протоколы обмена без деталей их реализации);
- аппаратный интерфейс для связи с внешними техническими устройствами (аппаратные средства, протоколы, драйверы).

Граничный класс может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<boundary>> в секции имени.

3.5. Отношения между классами

Классы вступают друг с другом в отношения. На диаграмме классов выделяют такие виды отношений: ассоциация, наследование, агрегирование и использование. При изображении конкретной связи ей можно сопоставить текстовую метку, документирующую имя этой связи и/или ее роль. Имя отношения должно быть уникально в своем контексте. Также связь может иметь по ее концам обозначения кратности (множественности) отношений между классами, некоторые из которых показаны ни рис. 6.

*	неопределенное множество
1..*	один или много
0..1	ноль или 1
1..8	от 1 до 8
12	точно 12
{2,3,5,7,11}	определенное множество

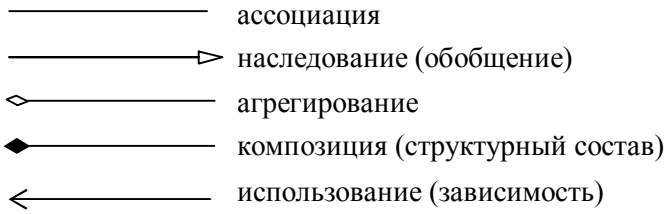


Рис. 6. Обозначения множественности и значки отношений между классами

Значок ассоциации соединяет два класса и означает наличие семантической связи между ними. Ассоциации часто отмечаются

существительными (например, Выбор), описывающими природу связи. Одна пара классов может иметь одну или несколько ассоциативных связей.

Наследование представляет отношение типа "общее/частное", выглядит как значок ассоциации со стрелкой, которая указывает от подкласса к суперклассу. При этом подкласс наследует структуру и поведение своего суперкласса. Класс может иметь один (одиночное наследование), или несколько (множественное наследование) суперклассов.

Агрегирование обозначает отношение "целое/часть" и получается из значка ассоциации добавлением закрашенного кружка на конце, обозначающем агрегат. Экземпляры класса на другом конце стрелки будут в определенном смысле частями экземпляров класса-агрегата. Разрешается рефлексивная и циклическая агрегация. Агрегация не требует обязательного физического включения части в целое.

Композиция – это разновидность агрегирования с четко выраженным отношением владения, причем время жизни частей и целого совпадают. Части с нефиксированной кратностью могут быть созданы уже после создания самого композита, живут и уничтожаются вместе с ним. Также части могут быть удалены явным образом еще до уничтожения композита. Кроме того, в композитном агрегировании целое отвечает за диспозицию своих частей, то есть композит должен управлять их созданием и уничтожением.

Использование обозначает отношение типа "клиент/сервер" и изображается как ассоциация с пустым кружком на конце, соответствующем клиенту. Эта связь выражает зависимость клиента от сервера и означает, что клиент нуждается в услугах сервера, то есть операции класса-клиента вызывают операции класса-сервера или имеют сигнатуру, в которой возвращаемое значение и/или аргументы принадлежат классу сервера. Фактически отношение использования показывает, что один элемент использует другой.

Общие рекомендации по построению диаграмм классов:

- использовать зависимость, только в случае, если моделируемое отношение не является структурным;
- использовать обобщение, только если имеет место отношение типа "является";
- располагать элементы так, чтобы свести к минимуму число пересекающиеся линий отношений;
- пространственно организовывать элементы так, чтобы семантически близкие сущности располагались рядом;
- чтобы привлечь внимание к важным особенностям диаграммы, использовать примечания (комментарии) и выделение цветом;

Требования к содержанию отчета

В подразделе 1 описывается назначение программы и цели ее создания. При этом следует руководствоваться рекомендациями, изложенными в разделе 2.2

настоящего пособия.

В подразделе 2 указывается перечень задач, программную реализацию которых предполагается осуществить. Разрабатывается статическая объектная модель будущей программы в виде диаграммы вариантов использования и диаграммы классов. Диаграммы реализовать в пакете программ Rational Rose, Visio или Altova UModel.

В подразделе 3 при формулировании функциональных требований к программе необходимо учитывать, что потребуется обеспечить их выполнение при программировании соответствующих задач.

Изложение функциональных требований разбивается на несколько пунктов в зависимости от количества задач, например:

3.1 Требования к задаче “...”

3.2 Требования к задаче “...”

3.3 Требования к задаче “...”

В каждом пункте указывается полное название задачи. Кратко излагается, какие в точности действия программа должна выполнить для реализации данной задачи, в форме словесного или формульно-словесного описания алгоритма. Кроме того, по каждой задаче могут быть указаны следующие требования:

– к временному регламенту реализации каждой задачи;
– к качеству реализации каждой задачи;
– к выходным данным;
– к входным данным;
– к преобразованию входной информации к машиночитаемому виду;
– к возможной одновременности выполнения нескольких задач;
– к достоверности результатов.

При описании требований к входным данным приводится: перечень и описание входных данных (идентификатор, форма представления, сроки и частота поступления), перечень и описание структурных единиц информации входных сообщений или ссылка источники этих данных. В описании для каждой структурной единицы информации входных сообщений следует указывать:

– наименование;
– необходимую точность ее числового значения (при необходимости);
– источник информации (документ, видеокадр, устройство, кодограмма, информационная база на машинных носителях и т.д.);
– идентификатор источника информации.

Требования к программной реализации задач должны содержать:

– требования к организации хранения данных в виде односвязного списка с заданной дисциплиной манипуляции данными (приложение А);
– предварительное описание (структуру) интерфейса для доступа к реализации отдельных задач и подзадач;
– требования к методу программирования, к структуре кода, к именованию программных и информационных объектов.

При формулировке специальных требований к математическому

обеспечению программной реализации задач приводят требования к составу, области применения (ограничение) и средств использования в системе математических методов и моделей, типичных алгоритмов и алгоритмов, которые подлежат разработке.

В требованиях к прикладному программному обеспечению указываются:

– требования к операционной среде;

– требования к инструментальным средствам программной инженерии, обеспечивающих разработку ПО (CASE-средства и средства объектно-ориентированного моделирования);

– требования к инструментальным средствам разработки ПО;

– требования к использованию готовых программных пакетов;

– требования к вспомогательным программным средствам (сервисные программы и утилиты).

В подразделе 4 указывают нефункциональные требования к разрабатываемой программе. При этом следует руководствоваться рекомендациями, изложенными в разделе 2.3 настоящего пособия. Следует учитывать, что реализация нефункциональных требований часто требует больших затрат, чем функциональных. Так, сопровождаемость требует значительных усилий по поддержанию соответствия проекта исходному коду и применения специальных методов создания модифицируемых программ и структур данных. Надежность – дополнительных средств восстановления системы при сбоях. Эффективность – поиска специальных архитектурных решений и оптимизации структуры системы и программного кода. А удобство – проектирования не «интуитивно» понятного, а профессионально понятного интерфейса пользователя. Все требования к подсистеме согласовываются с руководителем. ГОСТ на содержание технического задания на создание программ приведен в приложении Б.

Контрольные вопросы и упражнения

1. Что такое техническое задание и какова его структура?
2. Для чего и зачем разрабатываются компьютерные программы?
3. Какие положительные результаты могут быть получены в процессе использования компьютерной программы?
4. Что подразумевает программная реализация задач бизнес-процесса?
5. Каково назначение разрабатываемой компьютерной программы?
6. Назовите цели, в соответствии с которыми разрабатывается программа.
7. Дайте определение функционально-структурной и объектной модели компьютерной программы. Укажите принципиальные различия между этими моделями.
8. Перечислите базовые объектные модели компьютерной программы и компьютерные средства их реализации.
9. Что такое функциональные и нефункциональные требования к компьютерной программе?

10. Принципы формулировки требования к компьютерной программе при разработке технического задания.
11. Укажите основные аспекты формулирования функциональных требований к программной реализации задач бизнес-процессов.
12. Какие виды обеспечения компьютерной программы разрабатываются при ее создании?
13. Что такое прикладное обеспечение компьютерной программы?
14. Какие требования предъявляются к входным и выходным данным при программной реализации задач бизнес-процессов?
15. Какие требования предъявляются собственно к программной реализации задач бизнес-процессов?
16. Какие требования предъявляются к прикладному математическому обеспечению при программной реализации задач бизнес-процессов?
17. Улучшение каких технических, технологических, производственно-экономических или других показателей бизнес-процесса может быть достигнуто в результате создания и использования компьютерной программы?
18. Какова численность оперативного персонала, задействованного в автоматизируемых задачах до и после предполагаемого внедрения компьютерной программы?
19. Сформулировать функциональные и нефункциональные требования к программному приложению, реализующему поиск данных в односвязном списке тремя разными способами. Разработать объектные модели, описывающие работу программы.

Приложение А – Варианты индивидуальных заданий

Таблица А.1 Варианты индивидуальных заданий

№ вар.	Предметная область	Вид списка	Методы сортировки
1	2	3	4
1.	Обменный пункт: сотрудники пункта, виды валют, курсы валют, операции обмена.	Стек	1, 11, 21
2.	Ювелирный магазин: названия изделий, комитенты (кто сдал на комиссию), журнал сдачи изделий на продажу, журнал покупки изделий.	Очередь FIFO	2, 12, 20
3.	Поликлиника: врачи, пациенты, виды болезней, журнал учета визитов пациентов.	Очередь LIFO	3, 13, 19
4.	Кондитерский магазин: виды конфет, поставщики, торговые точки, журнал поступления и отпуска товара.	Дек	4, 14, 18
5.	Автобаза: автомашины, водители, рейсы, журнал выезда машин на рейсы.	Стек	5, 15, 1
6.	Парикмахерская: клиенты, прайс услуг, сотрудники, кассовый журнал.	Очередь FIFO	6, 16, 2
7.	Склад: поставщики товара, список товара, получатели товара, кладовщики.	Очередь LIFO	7, 17, 3
8.	Школа: учителя, предметы, ученики, журнал успеваемости.	Дек	8, 18, 4
9.	Оплата услуг на дачных участках: виды услуг, список владельцев, сотрудники управления, журнал регистрации оплат.	Стек	9, 19, 5
10.	Гостиница: проживающие, сотрудники гостиницы, номера, журнал регистрации проживающих.	Очередь FIFO	10, 20, 6
11.	Книжный магазин: авторы, книги, продавцы, покупатели, регистрация продаж.	Очередь LIFO	7, 1, 21
12.	Ремонтная мастерская: виды работ, исполнители, заказы на ремонт, заказчики.	Дек	8, 2, 20
13.	Аптечный киоск: номенклатура лекарств, работники аптеки, покупатели, журнал регистрации продаж.	Стек	9, 3, 19
14.	Выставка: стенды, стендисты, экскурсии, посетители.	Очередь FIFO	10, 4, 18
15.	Охранная служба: список постов охраны, список охранников, журнал выхода на дежурство, журнал учета замечаний.	Очередь LIFO	11, 5, 17
16.	Столовая: продукты, блюда, меню, журнал заказов	Дек	12, 6, 16

1	2	3	4
17.	Фото мастерская: заказчики работ, прайс работ, журнал поступления заказов, исполнители.	Стек	13, 7, 10
18.	Ветеринарная лечебница: список животных, список болезней, список хозяев, журнал посещений.	Очередь FIFO	14, 8, 11
19.	Сельское хозяйство: список растений, список угодий, список работников, журнал посевной.	Очередь LIFO	15, 9, 12
20.	Холдинг: список регионов, список предприятий, список показателей, журнал учета данных.	Дек	16, 10, 13
21.	Фонды предприятия: список основных средств, список категорий основных средств, список материально ответственных лиц, журнал учета состояния основных средств.	Стек	17, 11, 14
22.	Учет расхода материалов в компании: список статей затрат, список сотрудников, список отделов, журнал учета расхода материалов.	Очередь FIFO	18, 12, 15
23.	Фильмотека: список фильмов, список клиентов, список библиотекарей, журнал выдачи фильмов.	Очередь LIFO	19, 13, 9
24.	Цирк: список категорий артистов, список артистов, список цирковых площадок, журнал выхода артистов на работу.	Дек	20, 14, 8
25.	Спортивные заведения: список спортсменов, список видов спорта, список стадионов, журнал учета выступлений спортсменов.	Стек	21, 15, 7
26.	Компьютерные занятия: список слушателей курсов, список предметов, список преподавателей, журнал учета успеваемости.	Очередь FIFO	6, 16, 20
27.	Сбор урожая: список видов продукции, список сборщиков, список бригад, журнал учета урожая.	Очередь LIFO	5, 17, 19
28.	Фирма по обслуживанию населения: список заказчиков, список товаров, список разносчиков, журнал заказов.	Дек	4, 18, 15
29.	Партийная работа: список членов партии, список мероприятий, список городов, журнал учета выхода на мероприятие	Стек	3, 19, 16
30.	Экономическая база данных: список регионов, список показателей, список отраслей, отчетные статистические данные.	Очередь FIFO	2, 20, 17
31.	Журнальные статьи: список тем, список авторов, список названия статей, список журналов.	Очередь LIFO	1, 21, 18
32.	Анализ причин заболеваемости: список заболевших, список болезней, список районов, журнал учета заболевших.	Дек	1, 10, 14

1	2	3	4
33.	Отдел кадров: список сотрудников, штатное расписание, список отделов, журнал перемещения сотрудников по службе.	Стек	2, 11, 6
34.	Делопроизводство: список видов документов, карточка документа, список исполнителей, список департаментов	Очередь FIFO	3, 12, 7
35.	Расчет нагрузки на преподавателя: список преподавателей, список кафедр, предметов, журнал нагрузки.	Очередь LIFO	4, 13, 8
36.	Проектные работы: список проектов, список специалистов, список должностей, журнал учета работ.	Дек	5, 14, 1
37.	Учет компьютерного оборудования: список типов оборудования, список материально-ответственных лиц, список департаментов, журнал регистрации выдачи оборудования.	Стек	6, 15, 2
38.	Прививки детям: список прививок, список детей, список родителей, журнал учета прививок.	Очередь FIFO	7, 16, 3
39.	Начисление налогов в бюджет: виды налогов, список отраслей, список предприятий, журнал учета поступления налогов.	Очередь LIFO	8, 17, 4
40.	Экспертная система: список оцениваемых объектов, список экспертов, список регионов, журнал учета оценок.	Дек	9, 18, 5
41.	Ремонтная мастерская электрооборудования: список работ, список мастеров, список оборудования, список запасных частей, журнал учета выполненных работ.	Стек	10, 19, 9
42.	Магазин по продаже автомобилей: список фирм производителей, список автомобилей, журнал поступления автомобилей.	Очередь FIFO	11, 20, 8
43.	Автомобильный гараж: список владельцев, список автомобилей, список сторожей, журнал прибытия и убытия автомобилей.	Очередь LIFO	12, 6, 10
44.	Учет криминогенной ситуации в городе: список районов, список типов преступлений, список дежурных, журнал регистрации преступлений.	Дек	13, 7, 11
45.	Система здравоохранения: список регионов, список санаториев, список пенсионеров, журнал регистрации выдачи путевок в санатории.	Стек	14, 8, 20
46.	Туристические агентства: список туров, список стран, список клиентов, журнал регистрации продаж туров.	Очередь FIFO	15, 9, 19

1	2	3	4
47.	Продажа билетов на рейсы: список рейсов, прайс билетов, список компаний, журнал продаж билетов.	Очередь LIFO	16, 10, 18
48.	Продажа пиломатериалов: виды пиломатериалов, список заказчиков, журнал учета продаж пиломатериалов.	Дек	17, 1, 12
49.	Склад металлоконструкций: прайс товара металлоконструкций, список поставщиков, список сотрудников, журнал учета поставок.	Стек	18, 2, 13
50.	Система поддержки решений: список экспертов, список тем обсуждений, список департаментов, журнал учета предложений.	Очередь FIFO	19, 3, 17
51.	Детский сад: список родителей, список детей, список групп, журнал посещения детского сада.	Очередь LIFO	20, 4, 16
52.	Дом творчества молодежи: список кружков, список руководителей, список детей, журнал регистрации посещения кружков.	Дек	21, 5, 15

Таблица А.2 Методы внутренней сортировки данных

№ п/п	Метод сортировки
1	Сортировка простым выбором (выделением)
2	Сортировка обменом (пузырьком).
3	Улучшенный метод пузырька
4	Обменная сортировка со слиянием
5	Обменная сортировка с разделением
6	Обменная поразрядная сортировка
7	Сортировка вставками (включением)
8	Сортировка Шелла
9	Пирамидальная сортировка (сортировка кучей)
10	Быстрая сортировка
11	Сортировка подсчетом
12	Сортировка слиянием
13	Плавная сортировка
14	Сортировка перемешиванием
15	Распределяющая сортировка
16	Сортировка методом нахождения минимального элемента
17	Поиск перебором
18	Бинарный поиск
19	Сортировка с помощью включений с уменьшающимися расстояниями
20	Сортировка извлечением
21	Сортировка распределением

Приложение Б – Состав и содержание технического задания на создание программ (ГОСТ 34.602- 89)

№ п/п	Раздел	Содержание
1	Общие сведения	• полное наименование программы и ее условное обозначение • шифр темы или шифр (номер) договора; • наименование предприятий разработчика и заказчика системы, их реквизиты • перечень документов, на основании которых создается программа • плановые сроки начала и окончания работ • сведения об источниках и порядке финансирования работ • порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств
2	Назначение и цели создания (развития) программы	• вид автоматизируемой деятельности • перечень объектов, на которых предполагается использование программы • наименования и требуемые значения технических, технологических, производственно-экономических и др. показателей объекта, которые должны быть достигнуты при внедрении программы
3	Характеристика бизнес-процесса	• краткие сведения об объекте автоматизации • сведения об условиях эксплуатации и характеристиках окружающей среды
4	Состав и содержание работ по созданию программы	• перечень стадий и этапов работ • сроки исполнения • состав организаций — исполнителей работ • вид и порядок экспертизы технической документации • программа обеспечения надежности • программа математического обеспечения
5	Порядок контроля и приемки программы	• виды, состав, объем и методы испытаний программы • общие требования к приемке работ по стадиям • статус приемной комиссии
6	Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу программы в действие	• преобразование входной информации к машиночитаемому виду • изменения в объекте автоматизации • сроки и порядок комплектования и обучения персонала
7	Требования к документированию	• перечень подлежащих разработке документов • перечень документов на машинных носителях
8	Источники разработки	• документы и информационные материалы, на основании которых разрабатывается ТЗ и программы