Министерство образования и науки Кыргызской Республики

**Кыргызский Государственный Технический Университет**

**им. И. Раззакова**

Кафедра программного обеспечения компьютерных систем

**НАПРАВЛЕНИЕ ВПО 710400**

**«ПРОГРАММНАЯ ИНЖЕНЕРИЯ»**

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ №2**

**“ ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД К АНАЛИЗУ И ПРОЕКТИРОВАНИЮ”**

**к лабораторным работам**

**по дисциплине**

**ППР ПО**

кандидат технических наук, доцент

**Мусина Индира Рафиковна**

**Бишкек** **2015Содержание**

|  |  |  |
| --- | --- | --- |
| **№** | **Тема** | **Стр** |
| **1 (5)** | Формирование и анализ функциональных требований к ПО в MS VISIO | **3** |
| **2 (6)** | Алгоритмическое проектирование к ПО в MS VISIO | **11** |
| **3 (7)** | Анализ статической структуры проектируемой системы в MS VISIO | **18** |
| **4 (8)** | Анализ и моделирование взаимодействия объектов в MS VISIO | **41** |

**Лабораторная работа №5**

**Анализ функциональных требований к ПО в MS VISIO**

**Цель работы:** дать студентам навыки разработки и анализа функциональных требований с использованием диаграммы UML USE CASE

*Предоставить отчет с требований и представления диаграммы USE CASE, выполненных с использованием CASE-средства.*

*Примечание: диаграммы могут быть построены либо в MS VISIO, Rational Rose или System Architecture.*

1. **Изучение предметной области, формирование цели работы**

Наиболее важный вопрос, который следует задать, приступая к работе над новым программным продуктом, звучит следующим образом: «Какая именно система должна быть создана и зачем?».

Прежде чем взяться за проект, надлежит определить его назначение. Процесс формулирования целей будущей системы наряду с обдумыванием формы ее представления и общих ограничений составляет начальную фазу жизненного цикла проекта, которая завершается заявлением следующего вида: «Система, которую мы хотим создать, должна выполнять…» [1]. Для этого необходимо провести анализ бизнес-процессов в заданной предметной области, выявить проблемы.

Часто бывает полезно записать решаемую проблему в виде таблицы (Табл.1). Создание подобной таблицы является простым, но действенным средством, чтобы удостовериться в том, что все участники данного проекта работают вместе над осуществлением общей цели.

**Таблица 1. Структурирование проблемы**.

|  |  |
| --- | --- |
| **Элемент** | **Описание** |
| Проблема | Описание проблемы |
| Воздействует на что (кого) и результатом чего является | Указание лиц, на которых оказывает влияние данная проблема  Описание воздействия данной проблемы на заинтересованных лиц и бизнес-деятельность |
| Выигрыш от решения может состоять в следующем | Указание предлагаемого решения.  Список основных предоставляемых решением преимуществ. |

На основании проведенного системного анализа делается вывод о необходимости разработки программы, формулируются цели работы.

В качестве примера рассмотрим проблему учета университетских курсов.

Процесс регистрации студентов, желающих прослушать те или иные курсы, и распределения последних между профессорами зачастую весьма непрост и продолжителен.

После принятия профессорами решений о том, какие курсы они собираются преподавать на протяжении очередного семестра, служащий деканата вводит полученную информацию в компьютерную систему и распечатывает отчет, содержащий сведения о распределении педагогической нагрузки. Соответствующий каталог также публикуется и передается студентам.

Каждый студент заполняет многостраничную форму, отмечает выбранные курсы и возвращает форму в деканат. (Типичный студент в течение семестра посещает занятия по четырем курсам.) Данные из форм, полученных от студентов, также поступают в систему. Как только вся информация введена, выполняется процедура формирования учебного плана. В большинстве случаев первый вариант выбора, предъявленный студентом, оказывается и окончательным; если же возникают разногласия или несоответствия, студент приглашается в деканат, где его требования и предпочтения уточняются и заново учитываются. По завершении « увязки и утряски» студентам рассылаются «твердые» копии расписаний занятий. На весь процесс обычно уходит неделя, хотя в отдельных случаях он может растянуться и на две.

После окончания регистрации профессором передаются реестры студентов по каждому курсу.

Далее формулируется постановка задачи, которая в дальнейшем может уточняться. Приведем ее на примере системы учета университетских курсов.

1. Студенты в начале каждого семестра вправе запросить каталог, содержащий список курсов, которые предлагаются для изучения на протяжении этого семестра. Информация о каждом курсе, включающая его аннотацию, необходимые предварительные условия и фамилию профессора, помогает студентам принять осмысленные решения.

2. По окончании регистрации система учета курсов отсылает соответствующую информацию во внешнюю систему учета студентов.

3. Каждому профессору предоставляется возможность доступа к системе с целью выбора курсов, которые он хотел бы преподавать, и просмотра информации о том, кто из студентов предпочел эти курсы.

4. Семестру предшествует период времени, в течение которого студенты, обращаясь к системе, способны изменять расписания своих занятий путем добавления или удаления тех или иных курсов.

1. **Выявление и документирование требований к ПО**

Анализ и формулировка требований (Requirements Analysis and definition (RA&D)) представляет собой самый первый процесс, включающий в себя фактическую работу над разработкой ПО. В ходе данного процесса производится сбор всех необходимых требований по целевой системе как таковой, а также по ее окружению. После того, как будут собраны информация и требования от пользователя, следующий этап включает в себя их анализ. В ходе анализа идентифицируются действующие лица - люди, процессы или прочие системы, взаимодействующие с целевой системой, определяются ограничения. Ограничения – это ограничивающие факторы или значения для процесса, которые существуют в системе. Ограничения представляют собой очень важные предположения, при разработке и реализации системы. Все возможные ограничения должны обязательно выделяться и определяться с помощью опросов и анализа. Например, при разработке системы учета университетских курсов имеются следующие ограничения:

- разрабатываемая система должна позволить студенту выбрать четыре основных курса;

- аудитория каждого курса должна насчитывать не более десяти и не менее трех студентов. Курс, на который зарегистрировалось менее трех студентов, аннулируется. Поэтому каждый студент может указать два дополнительных курса на тот случай, если какие-либо из выбранных им основных курсов пользуются чрезмерной или недостаточной популярностью.

На завершающем этапе, после анализа собранной информации производится четкая формулировка перечня официальных требований к системе.

Процесс анализа и формулировки требований не нужно путать с системным анализом. В отличие от системного анализа в процессе анализа и формулировки требований не предусмотрено никаких консультаций. Процесс анализа и формулировки требований может, однако, включать в себя любые устные консультации, направленные на облегчение формулировки требований к целевой системе.

После выполнения процесса анализа и формулировки требований, как правило, должны быть получены следующие результаты: функциональные требования, модель требований, нефункциональные требования.

На начальной стадии формирования требований строится начальная диаграмма вариантов использования (Use case diagram).

**Варианты использования системы**

Case – анализ является наиболее эффективной методикой определения требований к системе, так как она концентрирует свое внимание на реальных действиях пользователя и тем самым уменьшает вероятность формулировки неверных требований с его стороны. Основной перечень процедур, выполняемых при проведении анализа вариантов использования, включает в себя следующее: выявление активных субъектов (actors), функций (вариантов использования) и связей между активными субъектами и вариантами использования разрабатываемой системы. Наиболее важной является коммуникативная составляющая модели, позволяющая группам разработчиков, заказчиков и конечных пользователей, обсуждающим свойства системы, говорить на одном языке.

Основы модели закладываются уже на начальной фазе процесса разработки, когда идентифицируются основные активные субъекты и варианты использования системы, а позже на этапе планирования, модель развивается и пополняется за счет уточнения существующих и добавления новых элементов.

1. *Определение действующих лиц (actors)*

Действующими лицами могут быть физические лица или внешние системы, которые будут взаимодействовать с создаваемой системой. Их можно идентифицировать, задавая следующие вопросы:

* + Кто использует систему непосредственно?
  + Кто вводит данные в систему?
  + Кто получает информацию из системы?
  + Кто отвечает за эксплуатацию системы?
  + Какие другие системы взаимодействуют с данной системой?

Каждый из внешних активных субъектов (actors) отождествляется с чем-то или кем-то, взаимодействующим с системой. Активный субъект способен выполнять различные функции:

* только вводить данные в систему;
* только получать информацию из системы;
* взаимодействовать с системой в обоих направлениях.

Процедура определения активных субъектов системы заслуживает особого внимания и обычно отличается итеративным характером – первый вариант списка субъектов редко бывает окончательным. Множество активных субъектов обычно обнаруживается уже в результате постановки задачи или в ходе обсуждения проблемы с потребителями или экспертами в предметной области. Рассмотрим выбор активных субъектов на примере системы учета университетских курсов. При этом может возникнуть следующий вопрос: является ли ранее зарегистрированный и новый студенты различными активными субъектами? Допустим, что изначально ответ на этот вопрос утвердительный. Далее следовало бы выяснить, каким именно образом субъекты - «студенты» обязаны взаимодействовать с системой. Если новый студент выполняет иные функции, нежели студент уже «известный» системе, речь, разумеется, должна вестись о различных активных субъектах. Если же, напротив, процесс использования системы студентами единообразен, все они отображаются в модели в виде одного и того же активного субъекта.

Другим примером может служить определение субъектов, соответствующих каждой функции, которые способен выполнять определенный пользователь. Решая подобную задачу, важно переусердствовать. Рассмотрим должность лаборанта кафедры университета. Вполне возможно, что лаборант ведет какой-либо курс, будучи студентом. Поскольку функции распределения педагогической нагрузки и выбора курсов для изучения уже закреплены в системе за активными субъектами «профессор» и «студент» соответственно, необходимость в «изобретении» нового субъекта вида «лаборант-студент», очевидно, не возникает.

Внимательно анализируя роли сущностей, взаимодействующих с системой, со временем нетрудно прийти к адекватному множеству активных субъектов.

В контексте системы учета университетских курсов могут быть получены следующие ответы на поставленные выше вопросы.

* Каждому студенту полагается выбрать курсы для изучения.
* Профессорам следует распределить между собой курсы, которые должны быть прочитаны в течение семестра.
* Служащие деканата обязаны создать каталог курсов и составить индивидуальные расписания работы студентов.
* Ответственность за администрирование базы данных со сведениями о профессорах курсах и студентах возлагается на деканат.
* Система учета курсов должна передавать информацию во внешнюю систему учета студентов.

На основании перечисленных ответов легко выделить следующие категории активных субъектов, взаимодействующих с системой учета университетских курсов: «студент», «профессор», «сотрудник деканата» и «Система учета студентов».

В модель следует включить краткое описание каждого активного субъекта, которое должно определять его роль в процессе использования системы. Описание активных субъектов системы учета университетских курсов могут выглядеть следующим образом:

* Студент – лицо, зарегистрированное в качестве слушателя университетских курсов;
* Профессор – лицо, наделенное полномочиями вести занятия в университете;
* Сотрудник деканата – служащий, ответственный за функционирование системы учета университетских курсов;
* Система учета студентов – внешняя система, обеспечивающая хранение информации о студентах.

1. *Варианты использования*

Вариант использования идентифицируется исходя из следующих соображений: каждый вариант использования представляет собой некоторую функцию, выполняемую системой в ответ на воздействие действующего лица, и характеризует конкретный способ применения системы, диалог между действующим лицом и системой. Набор вариантов использования системы охватывает множество заслуживающих внимания способов ее применения. Говоря формально, вариант использования – это последовательность выполняемых системой транзакций, которая приводит к получению некого ощутимого результата, в котором заинтересован определенный активный субъект.

Помощь в выборе вариантов использования системы окажут ответы на следующие вопросы.

* Какие задачи решает каждый активный субъект.
* Способен ли тот или иной субъект создавать, сохранять, изменять, удалять или считывать фрагменты данных в контексте системы?
* Какие варианты использования гарантируют выполнение указанных выше функций обработки данных?
* Должны ли активные субъекты сообщать системе о каких – либо непредвиденных внешних обстоятельствах?
* Имеет ли субъект право получать информацию об определенных событиях, происходящих в системе?
* Какие варианты использования связаны с поддержкой и администрированием системы?
* Удовлетворяются ли вариантами использования все функциональные требования, предъявляемые к системе?

При выборе варианта использования возникает вопрос: какие именно варианты считать «хорошими»? Одна из проблем связана с приемлемостью уровня детализации вариантов использования, т.е. с тем, насколько обширными (или узкими) они должны быть. Однозначного ответа нет. Однако лучше всего придерживаться следующего правила: «Вариант использования обычно охватывает изрядную долю функций, выполняемых системой от начала и до конца, и должен представлять для того или иного активного субъекта определенную ценность» [1]. Например, в системе учета университетских курсов субъект Студент обязан выбрать подходящие курсы, а информация о нем должна быть включена в расписания занятий и в систему учета студентов. Речь в данном случае идет о трех вариантах использования или только об одном? Скорее всего об одном., поскольку названные функции взаимозависимы: разве нормально, если студент, выбравший курсы, не будет упомянут в расписаниях или в системе учета студентов ( или хотя бы не получит уведомления о том, что сведения о нем не сохранены)? Если никто не зарегистрируется ни на один курс, то работа профессоров никому не нужна.

Еще одна проблема состоит в том, что делать с функциями, которые являются различными, но выглядят как родственные. Например, в обязанность сотрудника деканата входит добавление, модификация и удаление данных о курсах. Что это, три варианта использования или все-таки один? Целесообразнее оставить выбор на одном варианте, «ведение каталога курсов», поскольку эти функции выполняются одним активным субъектом «сотрудник деканата» и затрагивают одни и те же системные сущности (курсы).

Система учета университетских курсов обязана удовлетворять следующим требованиям.

* Активный субъект «студент» регистрирует курсы, которые он намерен прослушать.
* По завершении регистрации информация о студенте передается во внешнюю систему учета студентов.
* Субъект «профессор» использует систему для выбора курсов, которые он хотел бы преподавать в течение семестра, и для получения реестров студентов по каждому курсу.
* Субъект «сотрудник деканата» несет ответственность за ведение и опубликование каталога курсов на семестр и сохранение всей информации о расписаниях студентов и профессорах.

Варианты использования системы, идентифицированные на основе названных требований, перечислены ниже:

* Регистрация курсов студентом:
* Выбор курсов профессором;
* Получение реестров студентов по курсу;
* Сохранение информации о курсах;
* Сохранение информации о профессорах;
* Сохранение информации о студентах;
* Ведение каталога курсов.

Необходимо привести описание каждого варианта использования, выраженное в виде нескольких предложений. Вот как может выглядеть краткое описание варианта использования «регистрация курсов студентом»:

«Вариант использования инициируется активным субъектом Студент и предлагает возможности создания, просмотра и/или модификации индивидуального расписания на указанный семестр».

Подробную информацию о построении диаграмм вариантов использования смотрите в источниках [1], [4] – [6]. На рис.1 представлена основная диаграмма вариантов использования системы учета университетских курсов.

**Задание на самостоятельную работу:**

Сформировать требования и провести и их анализ для разработки систем, представленных в списке (задание выдается преподавателем каждому студенту, список прилагается).

**Используемые источники:**

1. Терри Кватрани. Визуальное моделирование с помощью Rational Rose 2002 и UML.Издательский дом «Вильямс». Москва – Санкт-Петербург - Киев. 2003
2. Вендров А. Практикум по проектированию программного обеспечения экономических информационных систем: Учеб. пособие. М.: Финансы и статистика, 2002. 192 с.
3. Маклаков С. BPwin и Erwin. CASE – средства разработки информационных систем. М.: Диалог-МИФИ, 2001. 304 с.
4. Мусина И.Р., Каримова Г.Т. Технология разработки программного обеспечения: Методические указания к практическим работам /Кырг. гос. техн. ун-т, Бишкек, 2007, 61с.
5. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. Пер.с англ.-М.:Издательство Бином, СПб, 1999.
6. Рамбо Дж., Якобсон А., Буч Г. UML: специальный справочник. СПб.: Питер, 2002. 656 с.

**Система учета**

**университетских курсов**

Студент

Система учета студентов

<<uses>>

<<uses>>

Рис.1 Система учета университетских курсов.

**Лабораторная работа №6**

Алгоритмическое проектирование к ПО в MS VISIO

**Цель работы:** дать студентам навыки разработки и анализа функциональных требований с использованием диаграммы UML USE CASE

*Предоставить отчет с требований и представления диаграммы USE CASE, выполненных с использованием CASE-средства.*

*Примечание: диаграммы могут быть построены либо в MS VISIO, Rational Rose или System Architecture.*

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической или логической реализации. Важно подчеркнуть то обстоятельство, что с увеличением сложности системы строгое соблюдение последовательности выполняемых операций приобретает все более важное значение.

Диаграммы действий воспроизводят поток функций управления, показывают, какие ветви процесса могут выполняться параллельно, и определяют альтернативные пути достижения целей. Диаграммы действий, конструируемые на начальных фазах жизненного цикла системы, представляют потоки, которые охватывают несколько вариантов использования или протекают на уровне определенного варианта. Позже, по мере детализации проекта, создаются и диаграммы действий, иллюстрирующие особенности реализации отдельных операций.

Элементами диаграммы действий служат собственно действия (activities), переходы (transitions) от одного действия к другому, точки принятия решений (decision points) и полосы синхронизации (synchronization bars). Диаграмма действий может быть разделена на зоны (swimlanes), каждая из которых обычно связан с определенным активным субъектом, ответственным за выполнение соответствующего подмножества действий.

**Состояние деятельности**

Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действий, а дугами – переходы от одного состояния действия к другому.

Именно диаграммы деятельности позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних деятельностей и действий. Основным элементом диаграммы является деятельность. Интерпретация этого термина зависит от той точки зрения, с которой стоится данная диаграмма. На концептуальной диаграмме деятельность – это некоторая задача, которую необходимо выполнить вручную или автоматизированным способом. На диаграмме, построенной в аспекте спецификации или реализации, деятельность представляет собой некоторый метод над классом. Рекомендуется в качестве простого действия использовать глагол с пояснительными словами (рис.6.1)

Разработать план проекта

**Рис.6.1.** Деятельность

**Начальное и конечное состояние**

Начальное состояние представляет собой такое состояние, которое не содержит никаких внутренних действий. В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме графической области, от которой начинается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка (Рис.6.2).

●

**Рис.6.2**. Начальное состояние

Конечное состояние представляет собой такое состояние, которое также не содержит никаких внутренних действий. В этом состоянии находится объект по умолчанию в конечный момент времени. Оно служит для указания на диаграмме графической области, от которой завершается процесс изменения состояний. Графически начальное состояние в языке UML обозначается в виде закрашенного кружка, помещенного в окружность (Рис.6.3).

⦿

**Рис.6.3.** Конечное состояние

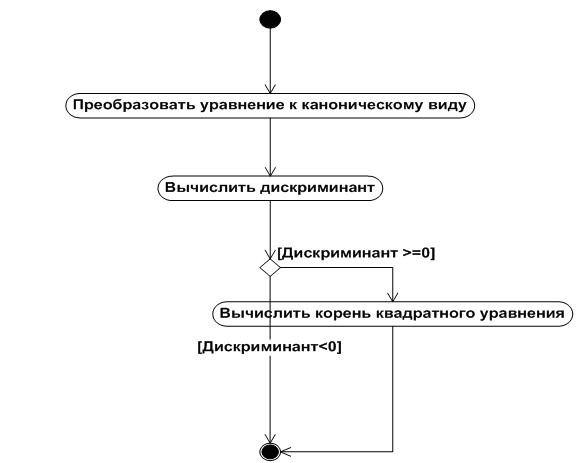
Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояния.

**Переходы**

Простой переход представляет собой отношение между двумя последовательными состояниями, которые указывают на факт смены одного состояния другим. Переход осуществляется при наступлении некоторого события: окончания выполнения деятельности. При построении диаграммы используются только нетригерные переходы, т.е. такие, которые срабатывают сразу после завершения деятельности или выполнения соответствующего действия и не связано с событием – триггером .Этот переход переводит деятельность в последующее состояние сразу, как только закончится действие в предыдущем состоянии. На диаграмме такой переход изображается сплошной линией со стрелкой.

Если из состояния действий выходит единственный переход, то он может быть никак не помечен. Если же таких переходов несколько, то сработать может только один из них. Именно в этом случае для каждого из таких переходов должно быть записано сторожевое условие в прямых скобках. При этом для всех выходящих из некоторого состояния переходов должно выполняться требование истинности только одного из них. Подобный случай встречается тогда, когда последовательно выполняемая деятельность должна разделиться на альтернативные ветви в зависимости от значения некоторого промежуточного результата. Такая ситуация получила название ветвления, а для ее обозначения применяется специальный символ. Графически ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста. В этот ромб может входить только одна стрелка от того состояния, после выполнения которого поток управления должен быть продолжен по одной из взаимно исключающих ветвей. Выходящих стрелок может быть две или более, но для каждой из них явно указывается соответствующее сторожевое условие в форме булевского выражения.

**Пример 1.** Фрагмент известного алгоритма нахождения корней квадратного уравнения (Рис.6.4).



**Рис.6.4.** Вычисление корня квадратного уравнения

Один из наиболее значимых недостатков обычных блок-схем или структурных схем алгоритмов связан с проблемой изображения параллельных ветвей отдельных вычислений. Поскольку распараллеливание вычислений существенно повышает общее быстродействие программных систем, необходимы графические примитивы для представления параллельных процессов. В языке UML для этой цели используется специальный символ для разделения и слияния параллельных вычислений или потоков управления - прямая черточка (Рис.6.5).

**а) разделение**

**б) слияние**

**Рис.6.5**

**Пример 2.** Для иллюстрации особенностей параллельных процессов выполнения действий рассмотрим ставший уже классическим пример с приготовлением напитка, представленного на рис.6.6.

На диаграмме представлены простые последовательности деятельностей (методов) для класса *Личность* (например, за деятельностью «Положить кофе в фильтр» следует деятельность «Вставить фильтр в автомат»), а также действия, содержащие условия (например, личность осуществляет деятельность «Поискать напиток», выбирая между кофе и колой).

Предположим, что мы отыскали кофе и идем вниз по «кофейному маршруту». Этот путь идет к линейке синхронизации, с которой связана активизация трех деятельностей: «Положить кофе в фильтр», «Добавить воду в емкость» и «достать чашки». Диаграмма указывает на то, что три деятельности могут выполняться параллельно. По существу это означает, что порядок их выполнения их деятельности не играет роли. Можно также выполнять эти деятельности, чередуя их друг с другом или одновременно. Если при описании поведения системы имеются параллельные деятельности, то их необходимо синхронизовать. Так, кофейный автомат не будет включаться до тех пор, пока в него не вставлен фильтр и не добавлена вода в емкость. Именно поэтому на диаграмме результаты этой деятельности сведены вместе к одной линейке синхронизации. Далее выполняется еще одна синхронизация: кофе должен быть готов и чашки должны стоять на месте перед тем, как мы сможем налить кофе.

Диаграммы деятельности могут быть использованы не только для спецификации алгоритмов вычислений или потоков управления в программных системах. Не менее важная область их применения связана с моделирование бизнес-процессы. Действительно, деятельность любой компании также представляет собой не что иное, как совокупность отдельных действий, направленных на достижение требуемого результата. Однако применительно к бизнес-процессам желательно выполнение каждого действия ассоциировать с конкретным подразделением компании. В этом случае подразделение несет ответственность за реализацию отдельных действий, а сам бизнес-процесс представляется в виде переходов действий из одного подразделения к другому. Для моделирования этих особенностей в UML используются **дорожки** (swimlanes).

 **Рис.6.6**

**Пример 3.** Ниже на рис.6.7 приведена диаграмма деятельности торговой компании, обслуживающей клиентов по телефону, с использованием дорожек.

**Пример 4.** На рис.6.8 представлена диаграмма действий для системы учета университетских курсов.

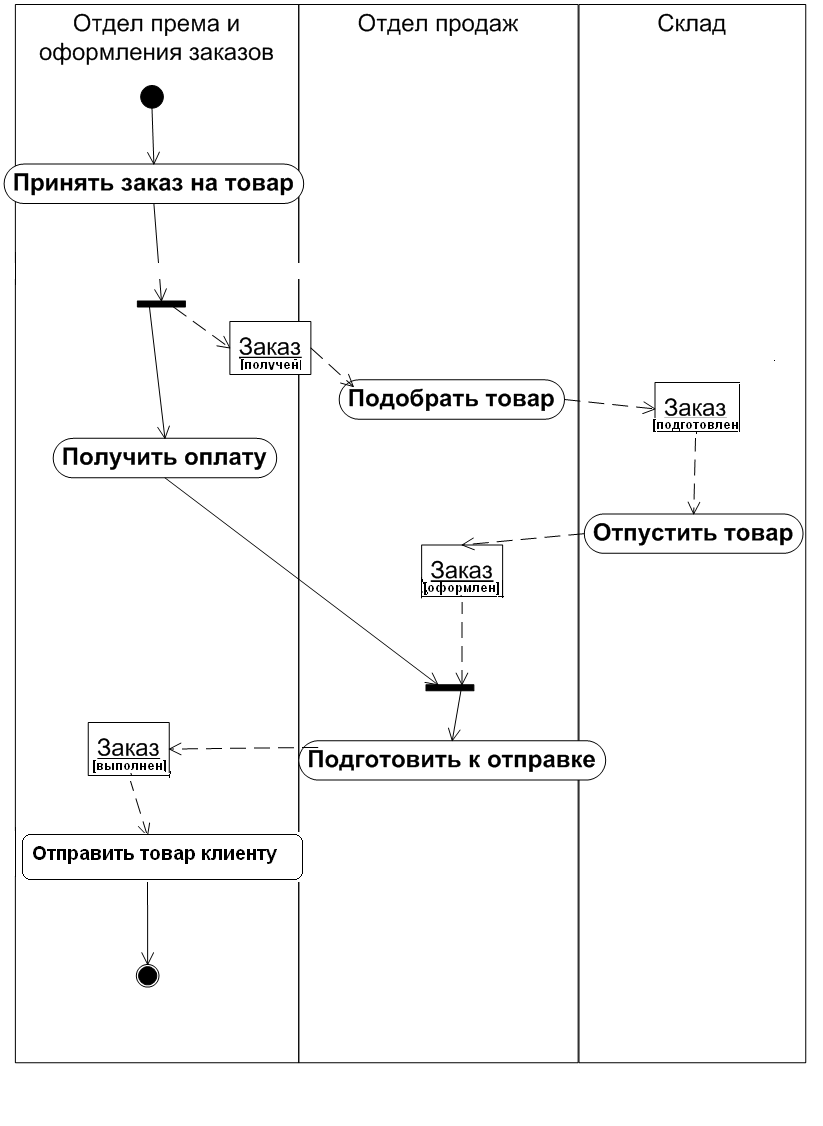


Рис.6.7



Рис.6.8 Диаграмма действий системы учета университетских курсов.

Далее требования должны быть уточнены. Графическое отображение системных требований для систем, ориентированных на данные, может быть проиллюстрировано с использованием диаграммы потоков данных DFD, которая является основным средством моделирования функциональных требований к системе. С ее помощью требования представляются в виде иерархии функциональных компонентов (процессов), связанных потоками данных. Главная цель такого представления - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами. Таким образом, модель системы в контексте DFD представляется в виде некоторой информационной модели, основными компонентами которой являются различные потоки данных, которые переносят информацию от одной подсистемы к другой.