2.4. Выделение подсистем

2.4.1. Понятие подсистемы

Итак, прикладная система представляет собой множество взаимозависимых объектов (см. п. 2.1). Каждый объект характеризуется набором атрибутов, значения которых определяют состояние объекта, и набором операций, которые можно применять к этому объекту. При разработке прикладных систем удобно считать, что все атрибуты объектов являются закрытыми (т.е. они не доступны вне объекта, и для того, чтобы в некотором объекте узнать значение атрибута другого объекта, или изменить его, необходимо воспользоваться одной из открытых операций этого объекта, если, конечно, такая операция определена). Операции объектов могут быть как открытыми, так и закрытыми.

Таким образом, каждый объект имеет строго определенный интерфейс, т.е. набор открытых операций, которые можно применять к этому объекту. Все объекты одного класса имеют одинаковый интерфейс. Интерфейс класса (а, следовательно, и каждого объекта этого класса) задается списком сигнатур его открытых (общедоступных) операций (и реализующих их методов); сигнатуры закрытых операций в интерфейс объектов соответствующего класса не входят.

Объектная модель системы задает множество взаимозависимых объектов, составляющих систему, и, следовательно, определяет набор интерфейсов, доступных внутри системы. Все возможности по обработке данных внутри системы (т.е. в каждом объекте, входящем в состав системы) определяются этим набором интерфейсов, который определяет внутреннее окружение (или среду) системы.

Наряду с внутренним окружением системы можно определить ее внешнее окружение. Оно определяется функциями (операциями), реализованными в составе системного программного обеспечения (т.е. операционной системы, системы программирования, различных редакторов, системы управления базами данных и т.п.), а также в других прикладных системах и библиотеках, используемых совместно с системой. Объекты и операции, составляющие внешнее окружение системы, тоже могут быть доступны внутри системы. Чтобы не упустить этого из виду, можно было бы добавить в объектную модель еще один объект, интерфейс которого представлял бы возможности внешнего окружения, используемые в системе (такой интерфейс обычно представляет лишь часть возможностей внешнего окружения). Но это было бы не совсем точно, так как внешнее окружение реализуется не одним, а несколькими объектами. С другой стороны внутри системы нет резона рассматривать структуру ее внешнего окружения. Выход из указанного противоречия во введении в рассмотрение еще одной сущности - подсистемы.

Подсистема - это набор объектов и подсистем, обеспечивающих некоторую функциональность, и взаимодействующих между собой в соответствии с их интерфейсами. Интерфейс подсистемы представляет собой подмножество объединения интерфейсов всех объектов и подсистем, составляющих эту подсистему. В состав подсистемы может входить один, или более взаимозависимых объектов и/или подсистем.

Множество интерфейсов объектов (и подсистем), которые в своей совокупности составляют некоторую подсистему, составляет внутреннее окружение этой подсистемы. В состав каждой подсистемы должна быть включена подсистема окружение, представляющая внешнее окружение этой подсистемы. Подсистема окружение для системы банковского обслуживания, рассматриваемой в качестве сквозного примера представлена на рисунке 2.41. Интерфейс подсистемы окружение определяет в каком программном окружении будет работать проектируемая система и какие возможности этого окружения будут

использоваться во время ее работы (это важно, когда возникает потребность модификации или замены отдельных компонентов окружения).

Отметим, что подсистема окружение представляет только интерфейс системы банковского обслуживания с ее внешним окружением. Внешнее окружение системы банковского обслуживания состоит из нескольких подсистем и библиотек, и для него тоже может быть разработана объектная модель, которая может содержать и разрабатываемую систему (в этой объектной модели она будет одной из подсистем).

Объектную модель системы банковского обслуживания и ее системного (внешнего) окружения тоже можно изобразить в виде объектной диаграммы (правда, в состав этой объектной диаграммы будут входить не объекты, а только подсистемы; каждая подсистема изображается на диаграмме в виде прямоугольника с двойными вертикальными сторонами). Зависимости между подсистемами, изображенные на этой объектной диаграмме (рисунок 2.42), отражают взаимодействие проектируемой системы банковского обслуживания и соответствующих подсистем в процессе работы системы. Тем самым определяются требования проектируемой системы к ее системному окружению.

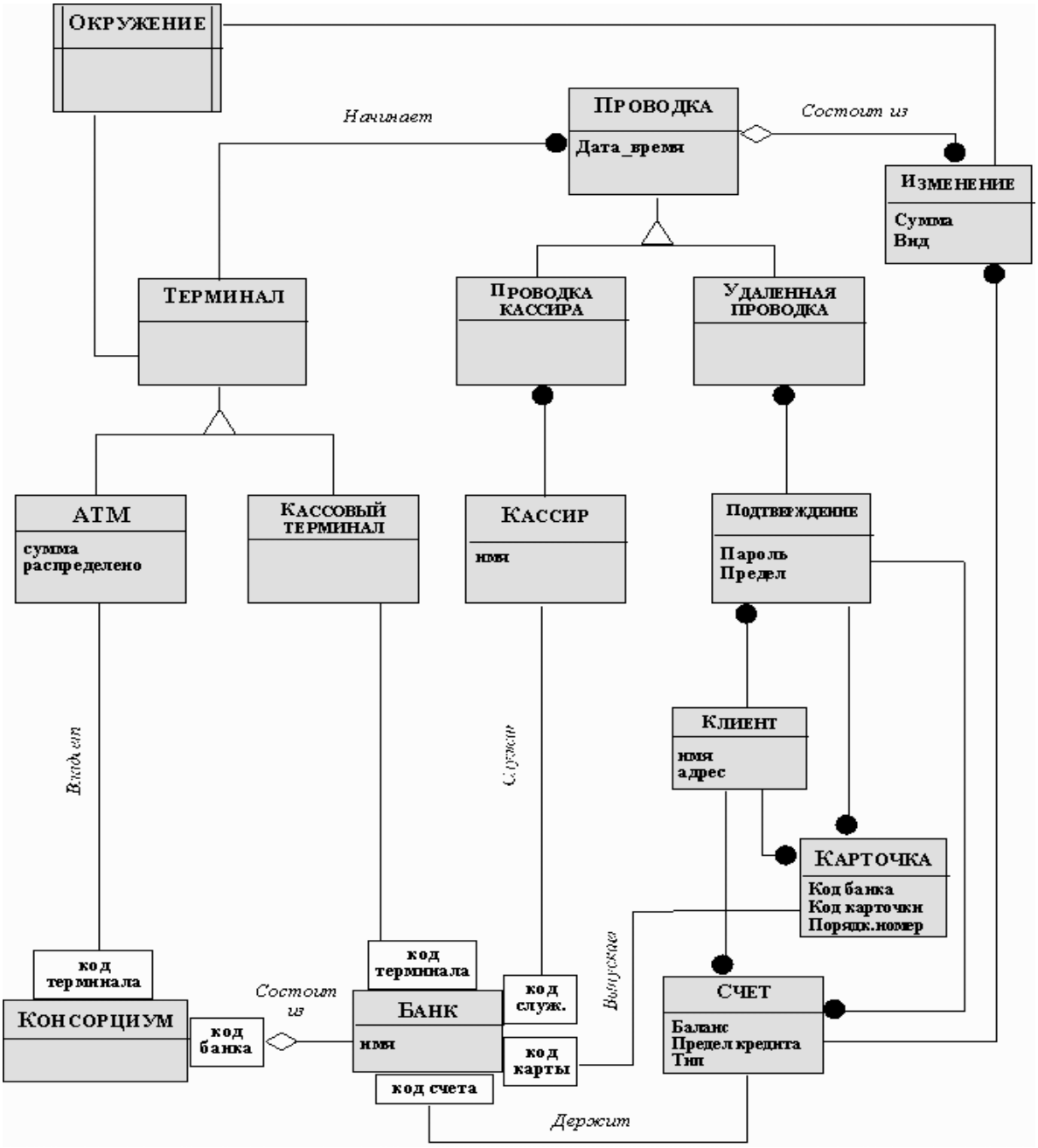

Рис. 2.41. Объектная диаграмма банковской сети, в которой указан интерфейс с системным окружением

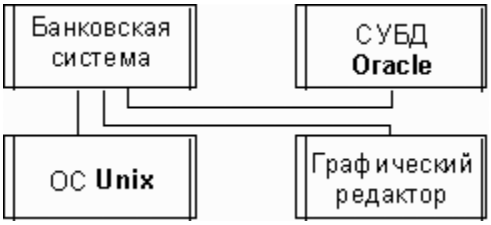

Рис. 2.42. Объектная диаграмма банковской сети и ее системного окружения

Введение понятия подсистемы и возможность включать в объектную модель наряду с объектами (классами) и подсистемы определяет иерархическую структуру объектной модели и позволяет использовать методологию OMT при проектировании достаточно сложных программных систем, содержащих большое число различных объектов и классов.

2.4.2. Интерфейсы и окружения

Объекты и подсистемы, составляющие подсистему более высокого уровня, будем называть компонентами последней. Как уже было отмечено, для каждого компонента, входящего в состав объектной модели подсистемы, определен его интерфейс, т.е. набор открытых (общедоступных) операций, которые можно применять к этому компоненту (объекту или подсистеме).

Интерфейс объекта определяется интерфейсом соответствующего класса и задается списком сигнатур его открытых операций (методов). Интерфейс подсистемы определяется через итерфейсы составляющих ее объектов и подсистем следующим образом: операция может быть включена в интерфейс подсистемы, если в составе этой подсистемы имеется объект (подсистема), интефейс которого содержит эту операцию. Интерфейсы описываются на языке описания интерфейсов IDL (Interface Definition Language).

Все возможности по обработке данных внутри подсистемы (т.е. в каждом компоненте, входящем в ее состав) определяются набором интерфейсов ее компонентов, который определяет внутреннее окружение подсистемы.

Если для некоторой подсистемы оказывается, что ни один ее компонент не содержит операции, которую желательно включить в ее интерфейс, в ее состав можно добавить объект, реализующий такую операцию. Такой объект называется интерфейсным объектом. Интерфейсные объекты позволяют согласовать внешний интерфейс подсистемы с ее внешним окружением, т.е. с интерфейсами других объектов и подсистем, которые вместе с рассматриваемой подсистемой составляют подсистему более высокого уровня.

Поясним введенные понятия на примере системы банковского обслуживания. В ее составе можно выделить подсистему банк (на самом деле в системе будет несколько экземпляров подсистемы банк - по одной для каждого банка, входящего в консорциум). При этом объектная модель системы примет вид, изображенный на рисунке 2.43.

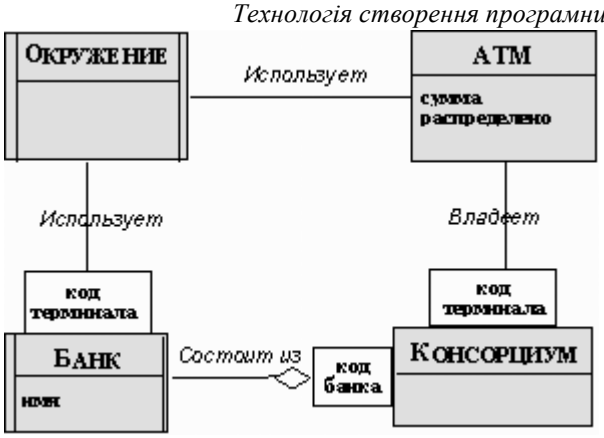

Рис. 2.43. Объектная диаграмма банковской сети после выделения подсистемы банк

При этом внешние интерфейсы подсистем банк и окружение вместе с интерфейсами объектов АТМ и консорциум образуют внутреннее окружение системы банковского обслуживания. Ее внешнее окружение представлено на рисунке 2.42; оно состоит из внешних интерфейсов различных программных систем, используемых в системе банковского обслуживания (на рисунке показана лишь часть этих систем), и ее собственного внешнего интерфейса.

2.5. Динамическая модель системы или подсистемы

Объектная модель представляет статическую структуру проектируемой системы (подсистемы). Однако знания статической структуры недостаточно, чтобы понять и оценить работу подсистемы. Необходимо иметь средства для описания изменений, которые происходят с объектами и их связями во время работы подсистемы. Одним из таких средств является динамическая модель подсистемы. Она строится после того, как объектная модель подсистемы построена и предварительно согласована и отлажена. Динамическая модель подсистемы состоит из диаграмм состояний ее объектов и подсистем.

2.5.1. События, состояния объектов и диаграммы состояний

Текущее состояние объекта характеризуется совокупностью текущих значений его атрибутов и связей. Во время работы системы составляющие ее объекты взаимодействуют друг с другом, в результате чего изменяются их состояния. Единицей влияния является событие: каждое событие приводит к смене состояния одного или нескольких объектов в системе, либо к возникновению новых событий. Работа системы характеризуется последовательностью происходящих в ней событий.

События

Событие происходит в некоторый момент времени (нередко оно используется для определения соответствующего момента времени). Примеры событий: старт ракеты, старт забега на 100 м, начало проводки (в банковской сети), выдача денег и т.п. Событие не имеет продолжительности (точнее, оно занимает пренебрежимо малое время).

Одно из событий может логически предшествовать другому, либо следовать за другим, либо они могут быть независимыми; примерами логически (причинно) связанных событий являются старт и финиш одного забега, начало проводки и выдача денег клиенту (в результате этой проводки), примерами независимых событий - старт ракеты и финиш забега, проводки, запущенные с разных АТМ, и т.п. Если события не имеют причинной связи (т.е.

они логически независимы), они называются независимыми (concurrent); такие события не влияют друг на друга. Независимые события не имеет смысла упорядочивать, так как они могут происходить в произвольном порядке. Модель распределенной системы обязательно должна содержать независимые события и активности.

События передают информацию с одного объекта на другой. Существуют классы событий, которые просто сигнализируют о том, что что-то произошло или происходит (примеры: загорание лампочки лифта, гудок в телефонной трубке). В программировании рассматриваются исключительные события (иногда их называют исключениями), которые сигнализируют о нарушениях работы аппаратуры, либо программного обеспечения.

Сценарии и трассы событий

Сценарием называется последовательность событий, которая может иметь место при конкретном выполнении системы. Сценарии могут иметь разные области влияния: они могут включать все события, происходящие в системе, либо только события, возникающие и влияющие только на определенные объекты системы.

На рисунке 2.44 приведен пример сценария пользования телефонной линией. Каждое событие в этом сценарии передает информацию с одного объекта на другой; например событие начинается длинный гудок передает сигнал от телефонной линии к вызывающему (пользователю). При анализе динамики работы системы необходимо составить и рассмотреть несколько сценариев, отражающих типичные варианты ее работы.

- ✓ вызывающий снимает трубку
- ✓ начинается длинный гудок
- ✓ вызывающий набирает цифру (9)
- ✓ гудок прекращается
- ✓ вызывающий набирает цифру (3)
- ✓ вызывающий набирает цифру (9)
- ✓ вызывающий набирает цифру ( )
- ✓ вызывающий набирает цифру ( )
- ✓ вызывающий набирает цифру ( )
- ✓ вызывающий набирает цифру ( )
- ✓ вызванный телефон начинает звонить
- ✓ вызывающий слышит гудки
- ✓ вызванный телефон отвечает
- ✓ гудки прекращаются
- ✓ телефоны соединены
- ✓ вызванный по телефону вешает трубку
- ✓ телефоны разъединены
- ✓ вызывающий вешает трубку

Рис. 2.44. Пример сценария: разговор по телефону

Следующим этапом после разработки и анализа сценариев является определение объектов, генерирующих и принимающих каждое событие сценария. Последовательности событий с привязкой к объектам проектируемой системы удобно представлять на диаграммах, называемых трассами событий. Пример трассы событий для разговора по телефону представлен на рисунке 2.45. Вертикальные линии изображают на этой диаграмме объекты, а горизонтальные стрелки - события (стрелка начинается в объекте, генерирующем событие, и заканчивается в объекте, принимающем событие). Более поздние события помещены ниже более ранних, одновременные - на одном уровне.

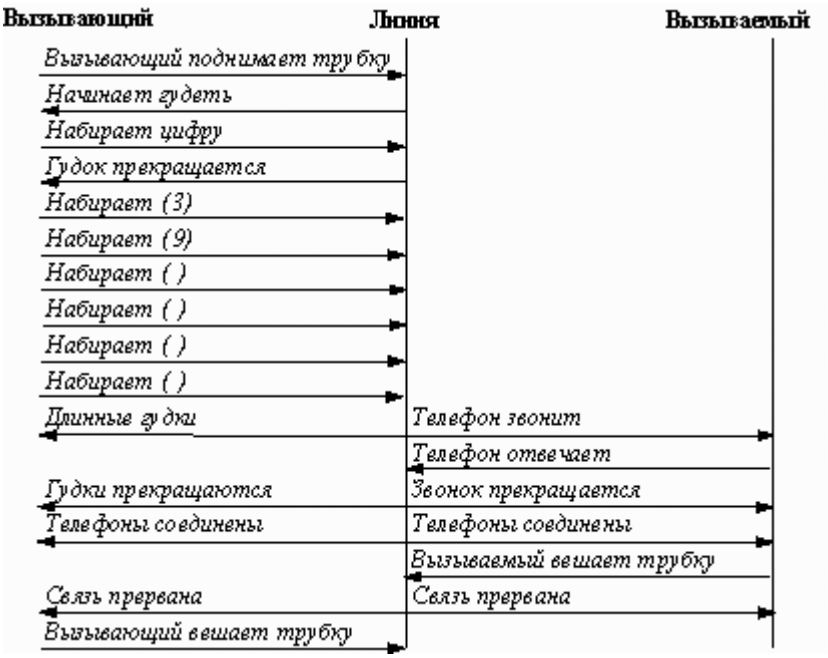

Рис. 2.45. Трасса событий для разговора по телефону

Состояния

Состояние определяется совокупностью текущих значений атрибутов и связей объекта. Например, банк может иметь состояния платежеспособный и неплатежеспособный (когда большая часть банков одновременно оказывается во втором состоянии, наступает банковский кризис).

Состояние определяет реакцию объекта на поступающее в него событие (в том, что реакция различна нетрудно убедиться с помощью банковской карточки: в зависимости от состояния банка обслуживание (реакция банка на предъявление карточки) будет разным). Реакция объекта на событие может включать некоторое действие и/или перевод объекта в новое состояние.

Объект сохраняет свое состояние в течение времени между двумя последовательными событиями, которые он принимает: события представляют моменты времени, состояния - отрезки времени; состояние имеет продолжительность, которая обычно равна отрезку времени между двумя последовательными событиями, принимаемыми объектом, но иногда может быть больше.

При определении состояний мы не рассматриваем тех атрибутов, которые не влияют на поведение объекта, и объединяем в одно состояние все комбинации значений атрибутов и связей, которые дают одинаковые реакции на события.

Диаграммы состояний

Диаграмма состояний связывает события и состояния. При приеме события следующее состояние системы зависит как от ее текущего состояния, так и от события; смена состояния называется переходом. Диаграмма состояний - это граф, узлы которого представляют состояния, а направленные деги, помеченные именами соответствующих событий, - переходы. Диаграмма состояний позволяет получить последовательность состояний по заданной последовательности событий.

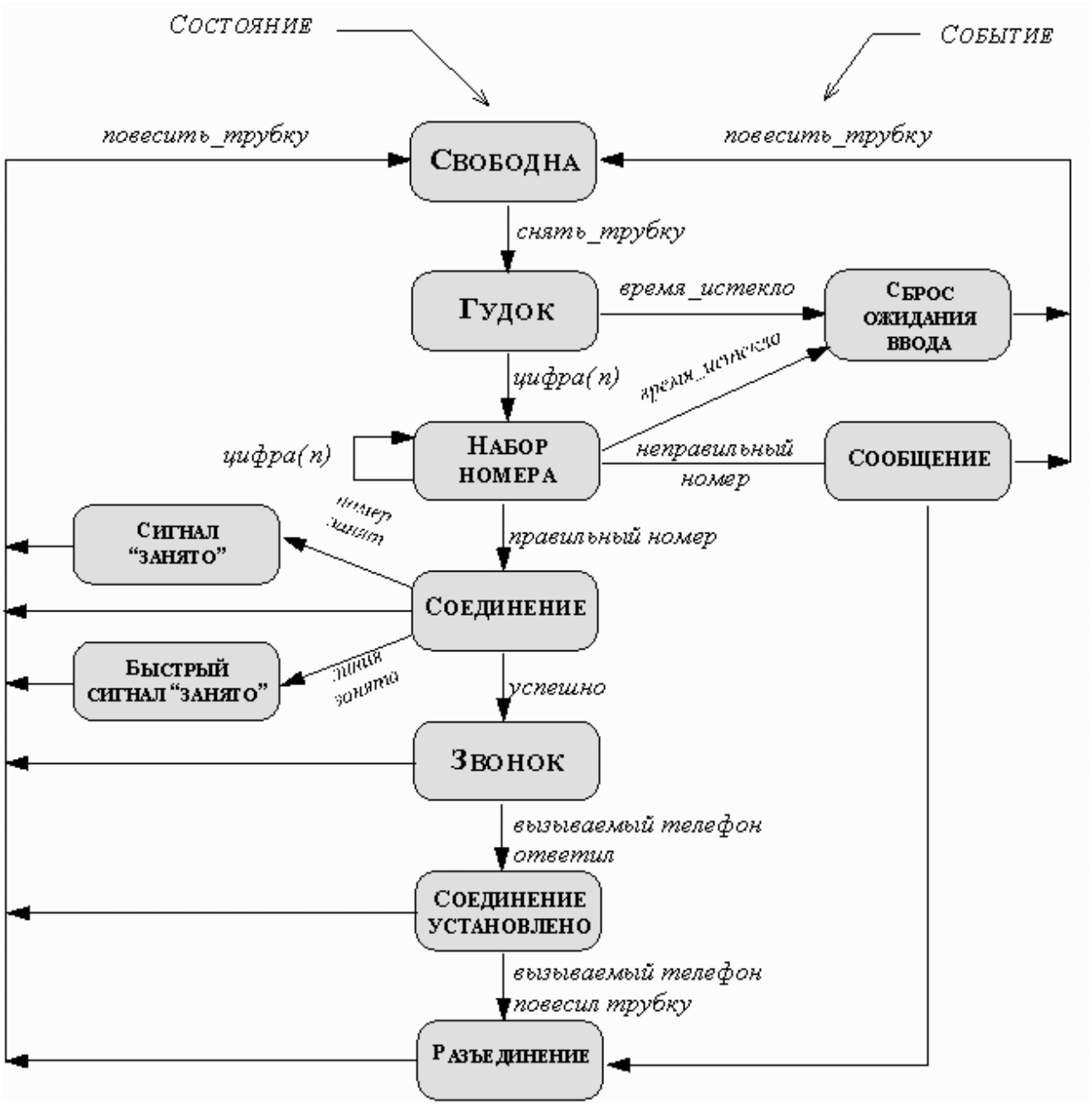

Рис. 2.46. Диаграмма состояний телефонной линии

На рисунке 2.46 приведена, в качестве примера, диаграмма состояний телефонной линии. Другие примеры диаграмм состояний см. в последующих пунктах.

2.5.2. Условия

Условие - это логическая (булева) функция от значений объектов, как например, карточку удалось прочитать, температура ниже нуля и т.п. Условие может выполняться в течение некоторого отрезка времени; событие, в отличие от условия, происходит мгновенно и не имеет продолжительности во времени.

Условия могут использоваться как ограничения на переходы: условный переход выполняется только тогда, когда и произошло соответствующее событие, и выполнено условие этого перехода (диаграмма состояний, представленная на рисунке 2.47, демонстрирует это на примере автомобильного движения по магистралям "Север-Юг" и

"Запад-Восток"). На диаграммах состояний условия записываются вслед за событиями в квадратных скобках.

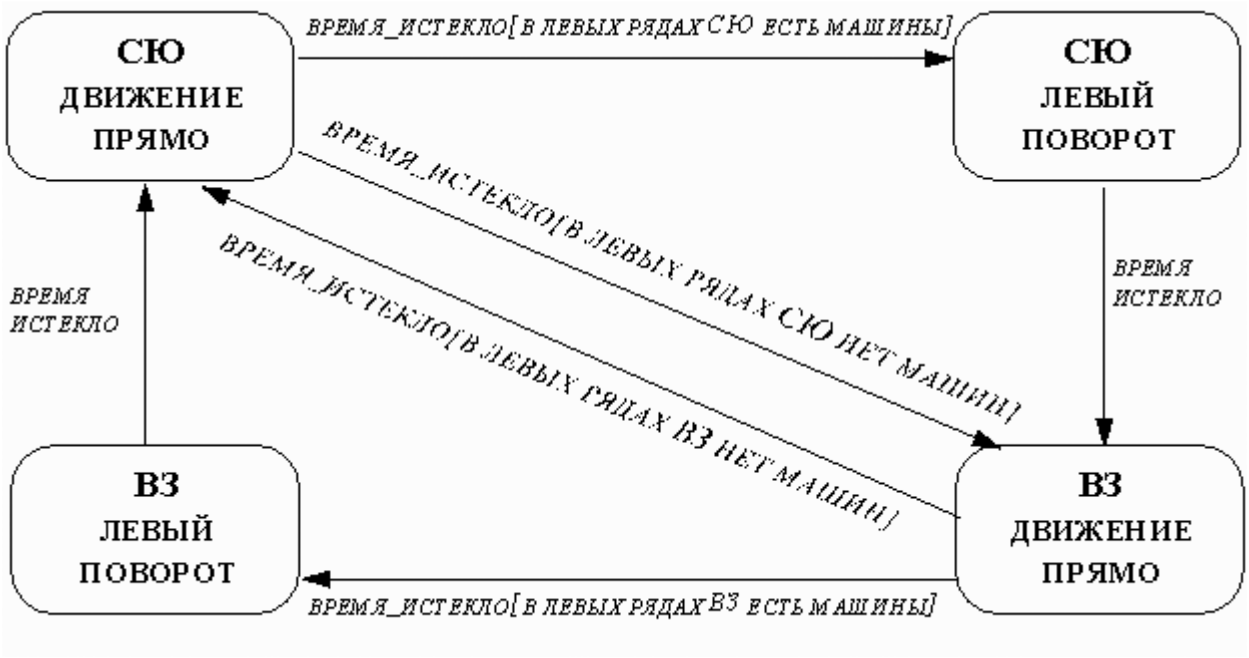

Рис. 2.47. Диаграмма состояний, на которой указаны условия

2.5.3. Активности и действия

Диаграмма состояний была бы не очень полезной, если бы она содержала только переходы (безусловные и условные), соответствующие генерируемым во время работы системы событиям. Являясь описанием поведения объекта, диаграмма состояний должна описывать, что делает объект в ответ на переход в некоторое состояние или на возникновение некоторого события. Для этого в диаграмму состояний включаются описания активностей и действий.

Активностью называется операция, связанная с каким-либо состоянием объекта (она выполняется, когда объект попадает в указанное состояние); выполнение активности требует определенного времени. Примеры активностей: выдача картинки на экран телевизора, телефонный звонок, считывание порции файла в буфер и т.п.; иногда активностью бывает просто приостановка выполнения программы (пауза), чтобы обеспечить необходимое время пребывания в соответствующем состоянии (это бывает особенно важно для параллельной асинхронной программы). Активность связана с состоянием, поэтому на диаграмме состояний она обозначается через "do: имя_активности" в узле, описывающем соответствующее состояние (см. рисунок 2.48).

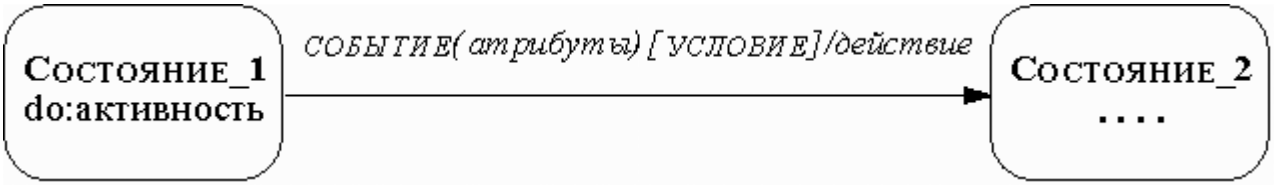

Рис. 2.48. Указание активностей и действий на диаграмме состояний

Действием называется мгновенная операция, связанная с событием: при возникновении события происходит не только переход объекта в новое состояние, но и выполняется действие, связанное с этим событием. Например, в телефонной сети событие повесил трубку сопровождается действием разъединить связь (см. рисунок 2.49). Действие указывается на диаграмме состояний вслед за событием, которому оно соответствует, и его имя (или описание) отделяется от имени события косой чертой ("/") (см. рисунок 2.48).

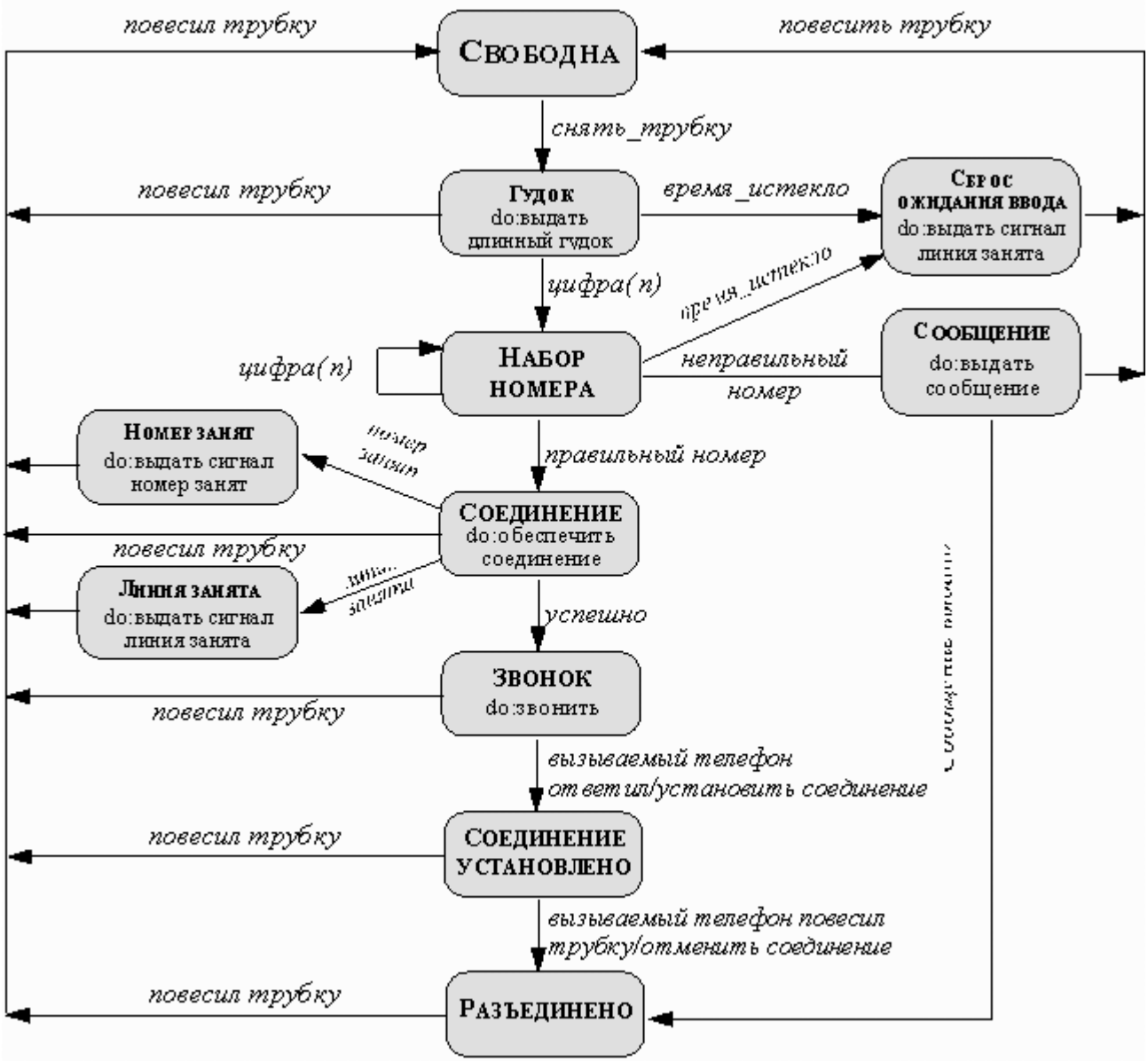

Рис. 2.49. Диаграмма состояний телефонной линии, на которой указаны активности и действия

Действия могут также представлять внутренние операции управления объекта, как например, присваивание значений атрибутам или генерация других событий.

2.5.4. Одновременные события. Синхронизация

На рисунке 2.50 представлена диаграмма состояний составного объекта: он состоит из четырех параллельно и независимо работающих компонентов (состояние каждого компонента не зависит от состояний остальных компонентов). Состояние такого объекта

определяется как кортеж, членами которого являются состояния каждого из составляющих объектов. Диаграмма состояний при этом распадается на четыре независимые диаграммы состояний каждого из компонентов.

В системах, состоящих из нескольких параллельно работающих объектов, иногда бывает необходимо согласовать (синхронизировать) работу двух или более объектов. Для этого тоже используются события, так как синхронизация по существу состоит в необходимости задержать переход одного из синхронизируемых объектов в очередное состояние, пока другой объект не придет в некоторое фиксированное состояние, а переходы из состояния в состояние управляются событиями. Синхронизирующее событие может вырабатываться в любом из синхронизируемых объектов, либо в каком-либо третьем объекте.

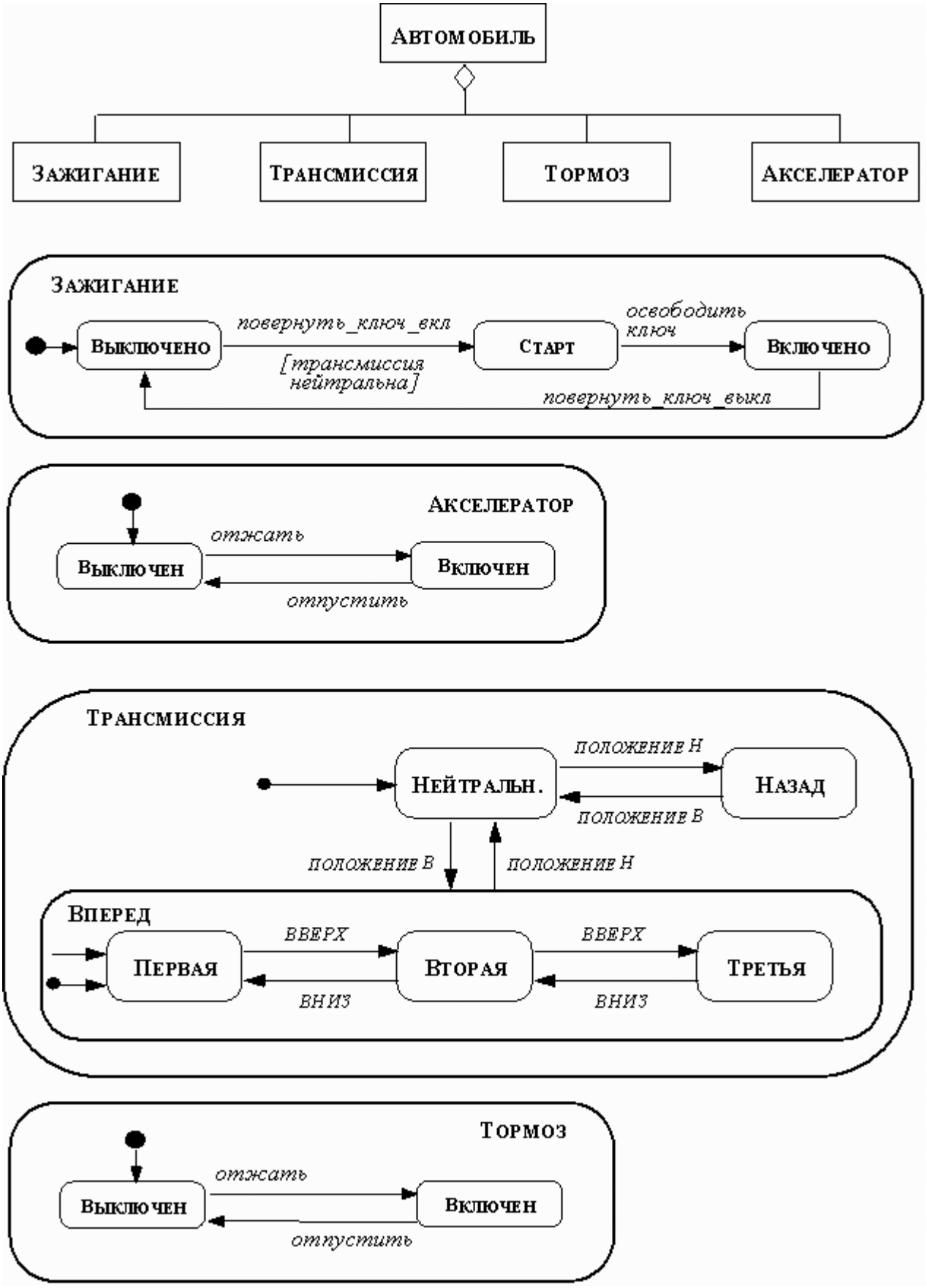

Рис. 2.50. Диаграмма состояний составного объекта (подсистемы)

Если синхронизирующее событие вырабатывается вне синхронизируемого объекта, оно может быть передано ему из другого объекта; на диаграмме это обозначается как показано на рисунке 2.51.

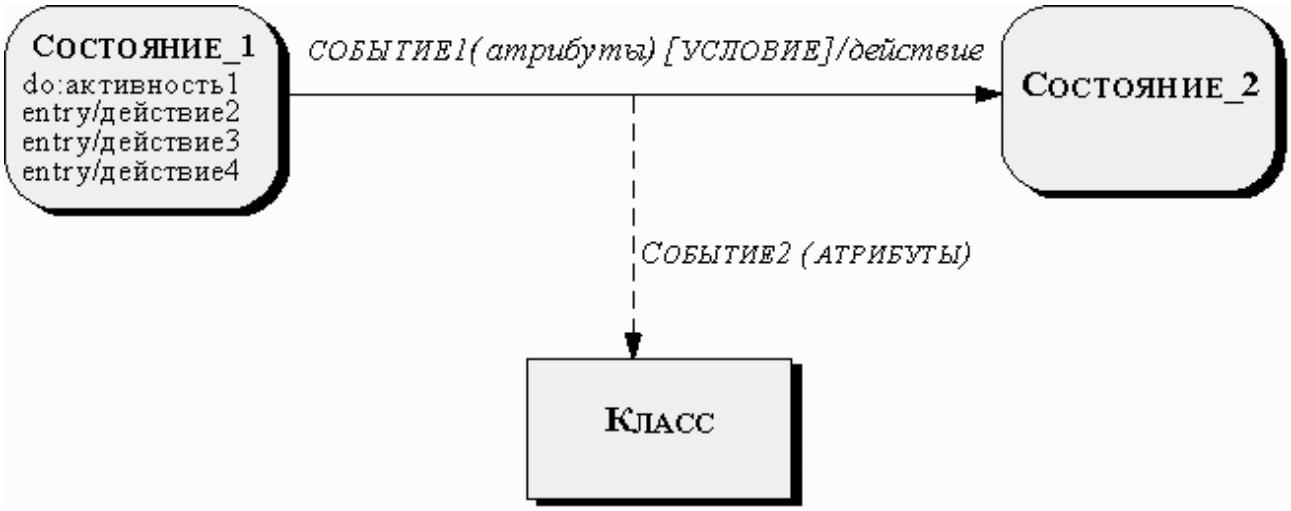

Рис. 2.51. Передача события из одного объекта другому

Если объект получает события из нескольких независимых объектов, то состояние, в которое он перейдет, может зависеть от порядка, в котором будут получены эти события (а этот порядок случаен, так как объекты независимы). Это называется условием конкуренции. Одной из задач проектирования является исключение нежелательных условий конкуренции. Это достигается с помощью синхронизации.

Синхронизация используется и в случае, когда в каком-либо состоянии требуется параллельно выполнить несколько активностей. В качестве примера рассмотрим устройство вывода АТМ (рисунок 2.52). Оно одновременно (параллельно) выдает наличные деньги и карточку; обе эти операции можно рассматривать как составную активность, состоящую из двух параллельно работающих активностей (пунктирная линия на диаграмме состояний делит состояние на две области, в каждой из которых выполняется одна из указанных активностей). Разделение управления на два параллельных потока схематически показано в виде стрелки, которая разделяется на две: событие готов вызывает переход из состояния установка сразу в два параллельных подсостояния состояния выдача; переход в следующее состояние происходит по двум событиям деньги взяты и карточка взята; если какое-либо из этих событий произойдет раньше другого, перехода все равно не будет, пока не произойдет и второе событие (в этом и состоит синхронизация).

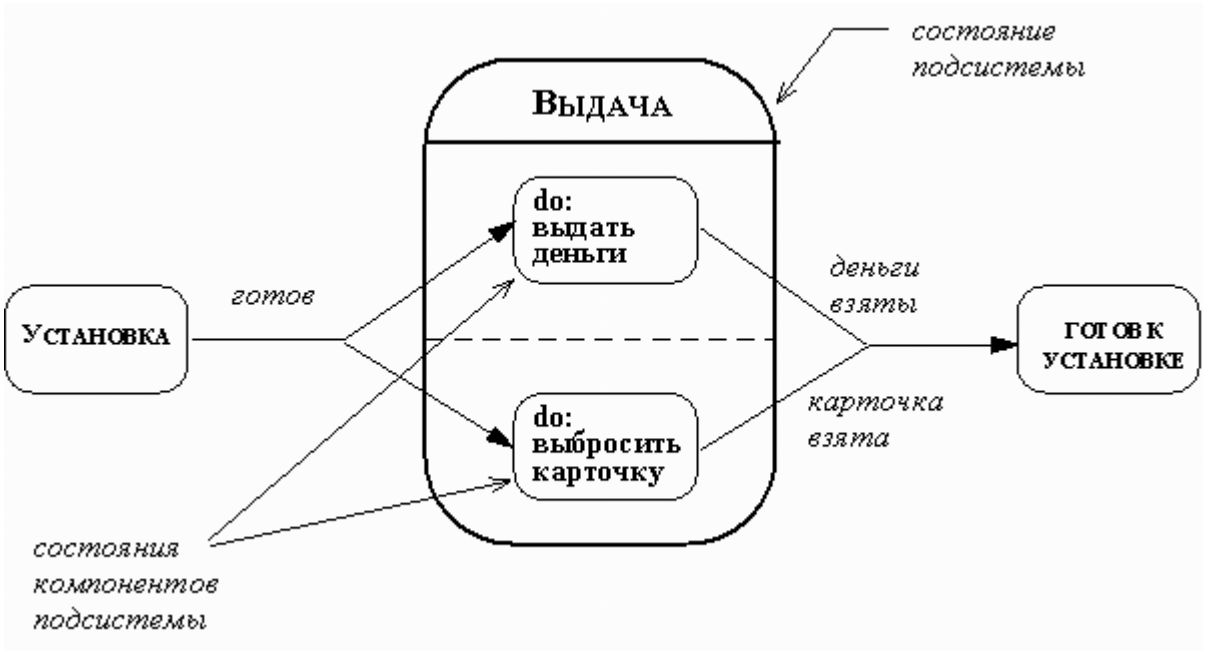

Рис. 2.52. Синхронизация в подсистеме

Из рассмотренного примера видно, что в различных состояниях может параллельно работать разное число процессов (активностей).

2.5.5. Вложенные диаграммы состояний

Диаграмма состояний компонента трансмиссия (рисунок 2.50) имеет узел (состояние) вперед, который сам является диаграммой состояний (вложенная диаграмма состояний). Интерпретация этой диаграммы следующая: трансмиссия имеет три состояния: нейтральное, назад и вперед; состояние вперед имеет три подсостояния (они уточняют состояние вперед): первая, вторая и третья. На диаграммах состояний суперсостояние изображается как прямоугольник с закругленными углами, внутрь которого помещаются все его подсостояния.

2.5.6. Динамическая модель банковской сети

В качестве примера применения рассмотренных принципов построения динамической модели построим динамическую модель банковской сети. Начнем с составления и изучения сценариев.

АТМ просит клиента вставить карточку
клиент вставляет карточку
АТМ принимает карточку и читает ее номер
АТМ просит ввести пароль
клиент вводит "1234."
АТМ передает номер и пароль в консорциум,
консорциум проверяет номер и пароль,
определяет код банка - "39" и сообщает АТМ, что запрос принят
АТМ просит клиента (с помощью меню на экране) выбрать вид проводки
(снятие, вклад, перевод, запрос)
клиент выбирает снятие
АТМ спрашивает клиента какова требуемая сумма

клиент вводит $100
АТМ убеждается, что введенная сумма в пределах лимита и
просит консорциум выполнить проводку,
консорциум передает запрос в банк,
банк выполняет проводку и возвращает новое значение баланса счета
АТМ выдает сумму и просит клиента взять ее
клиент берет деньги
АТМ спрашивает не нужно ли клиенту чего еще
клиент вводит нет
АТМ печатает счет, выдает карточку и просит клиента взять их
клиент берет счет и карточку
АТМ просит (другого) клиента ввести карточку

Рис. 2.53. Нормальный сценарий для банковской сети

На рисунке 2.53 представлен нормальный сценарий обслуживания клиента в банковской сети; один из возможных сценариев, содержащих исключительные ситуации, показан на рисунке 2.54.

АТМ просит клиента вставить карточку
клиент вставляет карточку
АТМ принимает карточку и читает ее номер
АТМ просит ввести пароль
клиент вводит "9999."
АТМ передает номер и пароль в консорциум;
консорциум, проконсультировавшись с соответствующим банком,
отвергает запрос
АТМ сообщает, что пароль введен неверно
клиент вводит "1234."
АТМ передает номер и пароль в консорциум,
консорциум проверяет номер и пароль,
определяет код банка - "39" и сообщает АТМ, что запрос принят
АТМ просит выбрать вид проводки
клиент выбирает снятие
АТМ спрашивает какова требуемая сумма
клиент (раздумав брать деньги) набирает отмену
АТМ выдает карточку и просит клиента взять ее
клиент берет карточку
АТМ просит (другого) клиента вставить карточку

Рис. 2.54. Сценарий для банковской сети, содержащий исключительные ситуации

Для каждого сценария можно составить соответствующую трассу событий (рисунок 2.55). Для этого выделяем в сценарии имена событий (событиями являются все сигналы, вводы данных, решения, прерывания, переходы и действия, выполняемые клиентом или внешними устройствами), указывая для каждого события объект, порождающий это событие (активный объект).

Имея трассы событий, можно построить диаграммы состояний объектов проектируемой системы. Банковская сеть есть агрегация определенного числа параллельно и независимо работающих объектов четырех классов: консорциум, банк, АТМ (банкомат) и клиент; поэтому состояние банковской сети определяется как кортеж состояний составляющих ее объектов: 1 объекта класса консорциум, b объектов класса банк, a объектов класса АТМ

(банкомат) и с объектов класса клиент (a, b, c - количество АТМ, банков и клиентов сети соответственно). Классификация объектов, используемая при объектно-ориентированном подходе, позволяет вместо a+b+1 диаграмм состояний построить всего три (диаграммы состояний клиентов строить не нужно, так как их текущее состояние ясно и так).

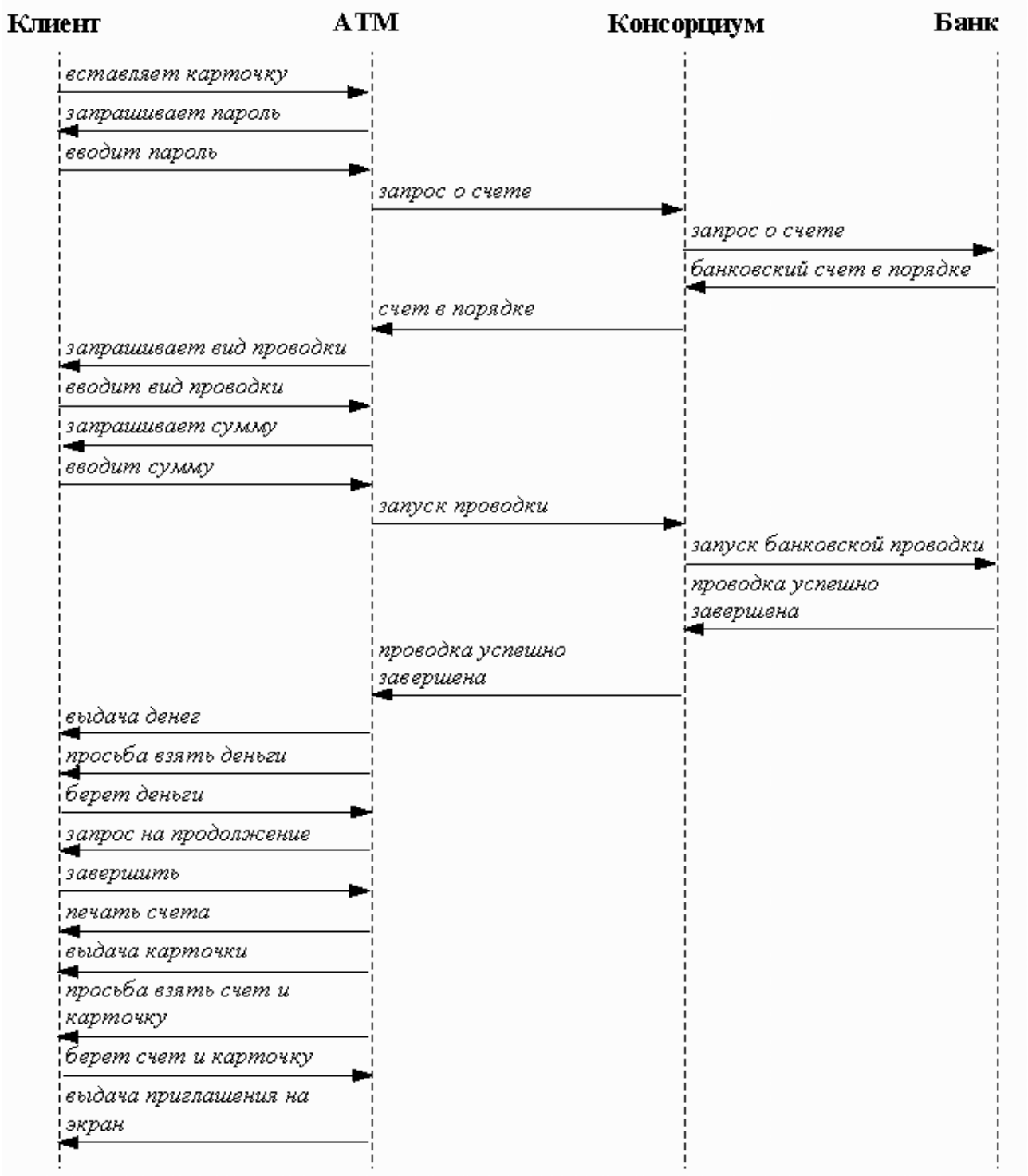

Рис. 2.55. Трасса событий в банковской сети

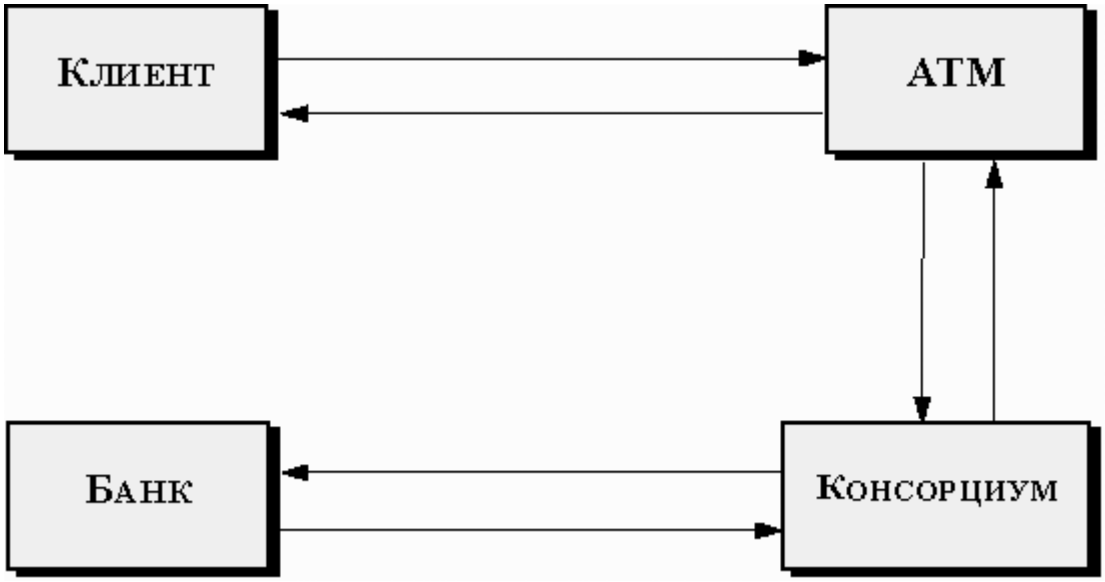

Рис. 2.56. Привязка событий к объектам банковской сети

Построение диаграмм состояний начинается с привязки событий к объектам банковской сети (см. рисунок 2.56), являющимся источниками этих событий. Сначала рассматриваются нормальные события, потом исключительные события. Построение диаграммы состояний объекта (класса) может считаться законченным, когда диаграмма охватывает все рассматриваемые сценарии. Диаграммы состояний объектов классов АТМ (банкомат), консорциум и банк представлены на рисунках 2.57, 2.58 и 2.59 соответственно.

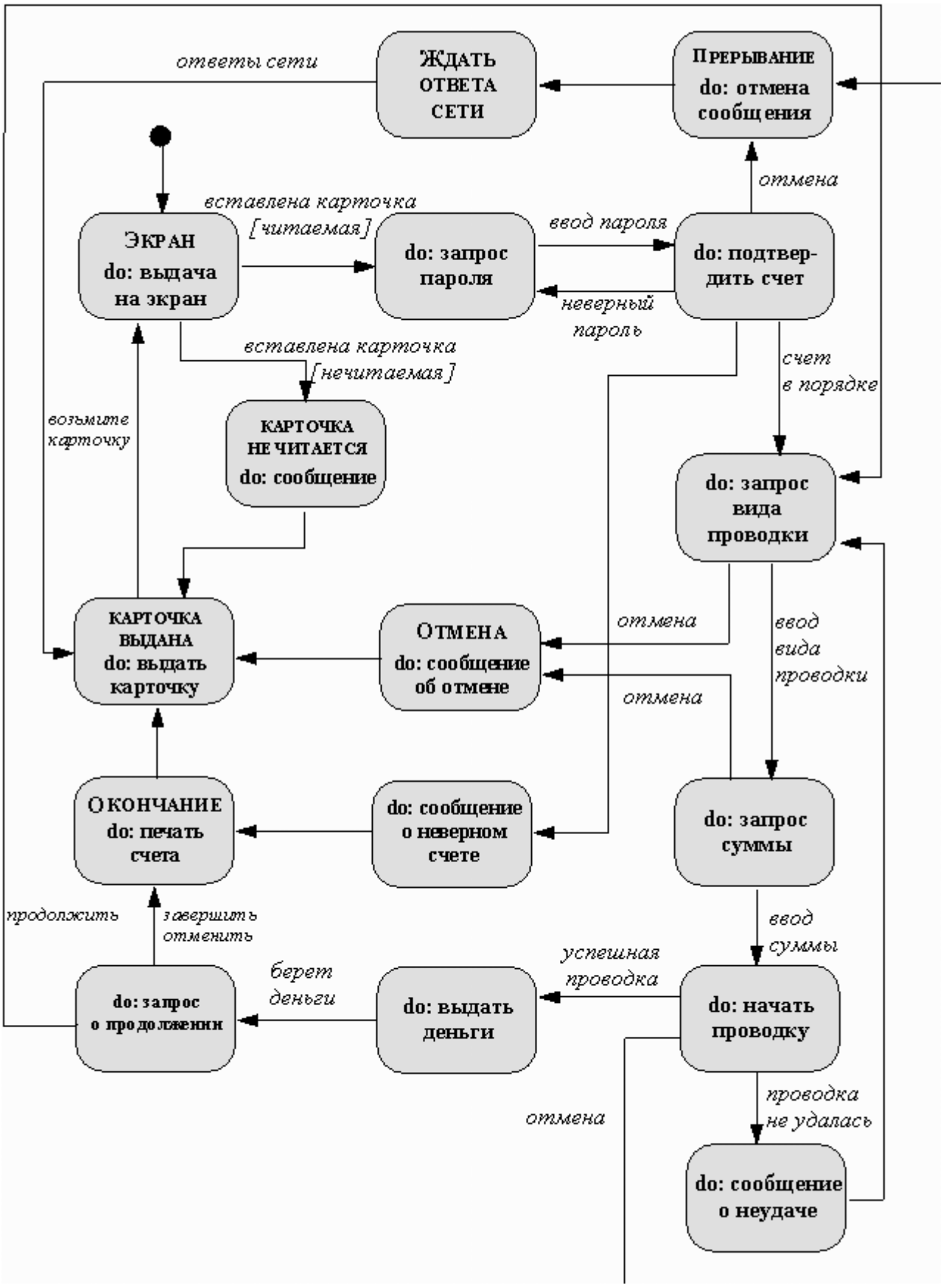

Рис. 2.57. Диаграмма состояний объектов класса ATM (банкомат)

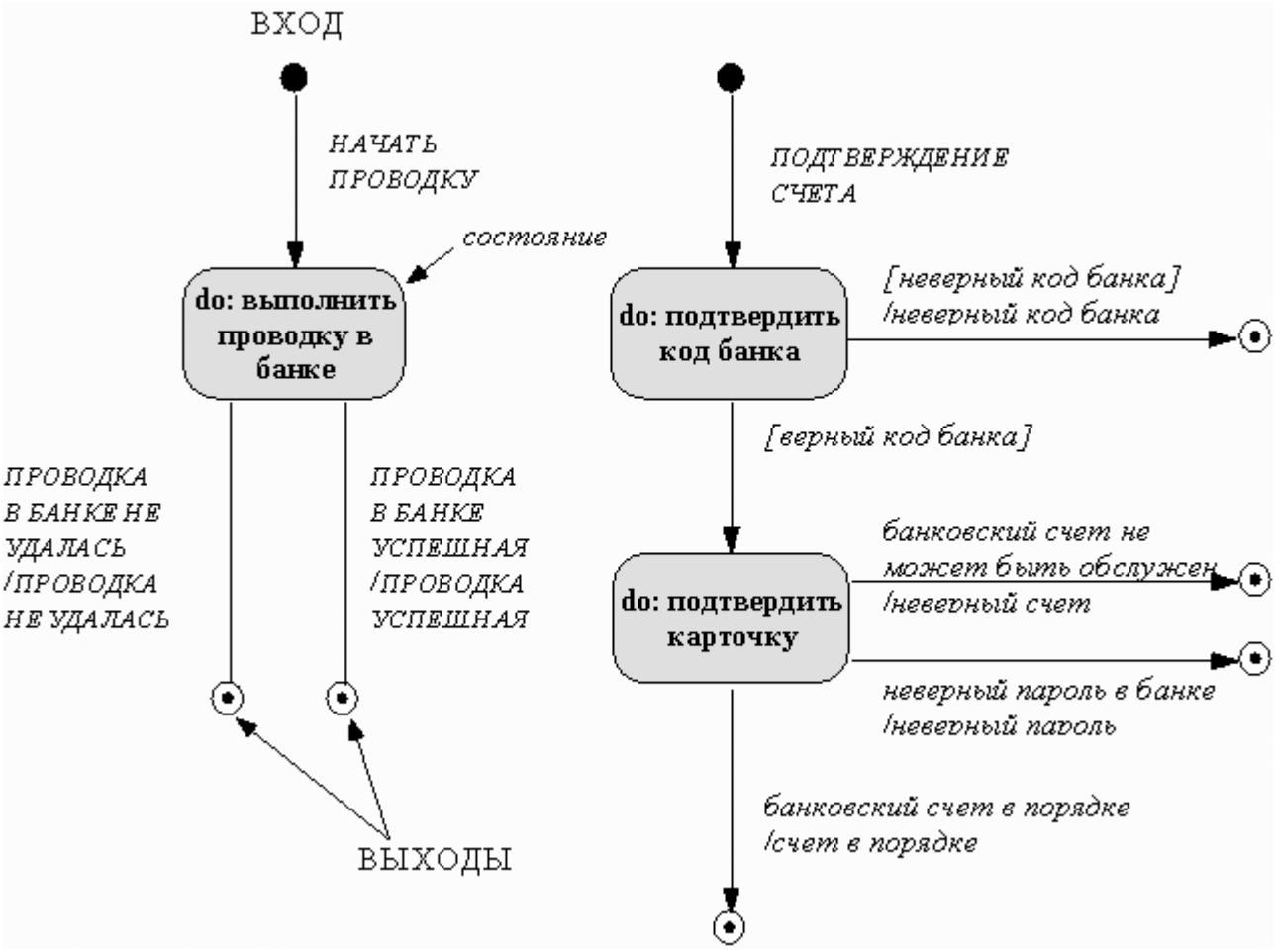

Рис. 2.58. Диаграмма состояний объектов класса консорциум

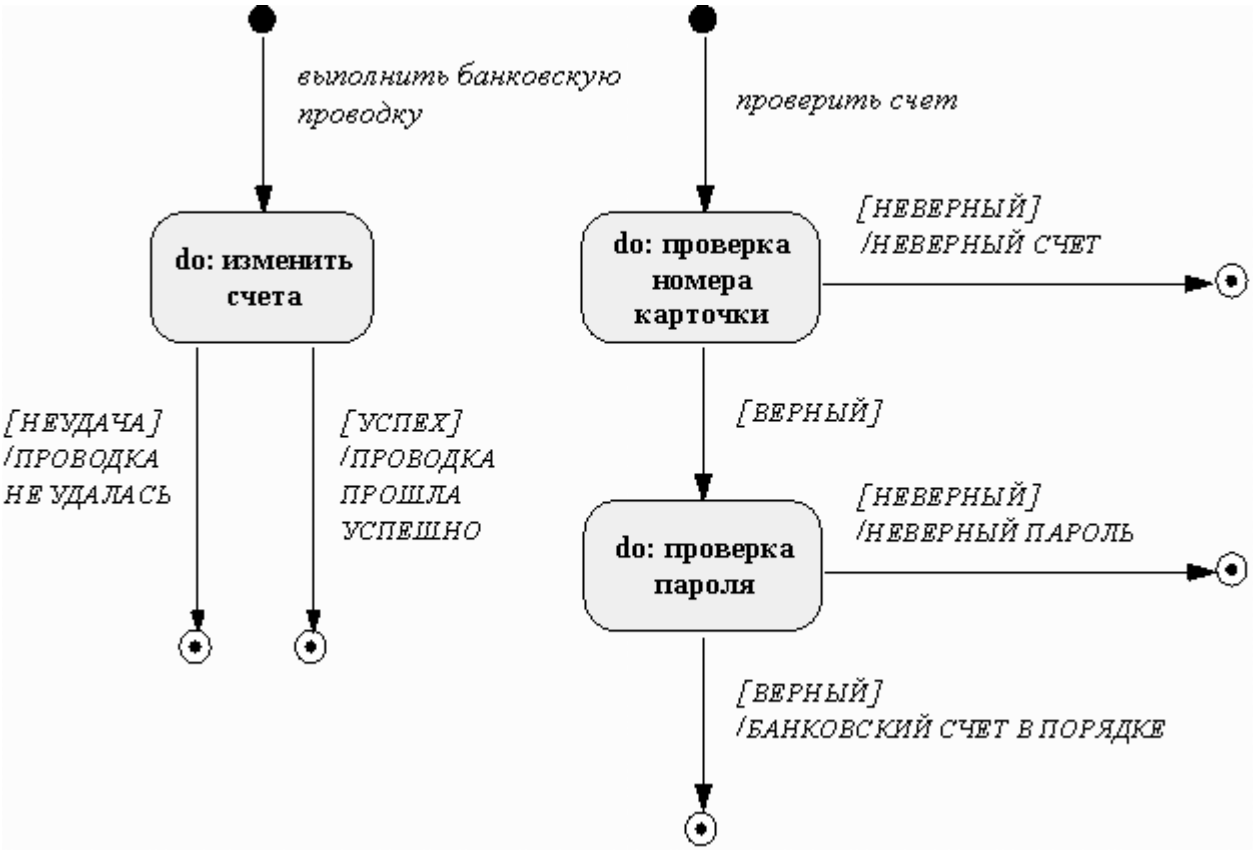

Рис. 2.59. Диаграмма состояний объектов класса банк

2.6. Функциональная модель подсистемы

Функциональная модель описывает вычисления в системе. Она показывает, каким образом выходные данные вычисляются по входным данным, не рассматривая порядок и способ реализации вычислений. Функциональная модель состоит из набора диаграмм потока данных, которые показывают потоки значений от внешних входов через операции и внутренние хранилища данных к внешним выходам. Функциональная модель описывает смысл операций объектной модели и действий динамической модели, а также ограничения на объектную модель. Неинтерактивные программы (например, компиляторы) имеют тривиальную динамическую модель: их цель состоит в вычислении значения некоторой функции. Основной моделью таких программ является функциональная модель (хотя если программа имеет нетривиальные структуры данных, для нее важна и объектная модель).

2.6.1. Диаграммы потоков данных

Функциональная модель представляет собой набор диаграмм потоков данных (далее - ДПД), которые описывают смысл операций и ограничений. ДПД отражает функциональные зависимости значений, вычисляемых в системе, включая входные значения, выходные значения и внутренние хранилища данных. ДПД - это граф, на котором показано движение значений данных от их источников через преобразующие их процессы к их потребителям в других объектах.

ДПД содержит процессы, которые преобразуют данные, потоки данных, которые переносят данные, активные объекты, которые производят и потребляют данные, и хранилища данных, которые пассивно хранят данные.

Процессы

Процесс преобразует значения данных. Процессы самого нижнего уровня представляют собой функции без побочных эффектов (примерами таких функций являются вычисление суммы двух чисел, вычисление комиссионного сбора за выполнение проводки с помощью банковской карточки и т.п.). Весь граф потока данных тоже представляет собой процесс (высокого уровня). Процесс может иметь побочные эффекты, если он содержит нефункциональные компоненты, такие как хранилища данных или внешние объекты.

На ДПД процесс изображается в виде эллипса, внутри которого помещается имя процесса; каждый процесс имеет фиксированное число входных и выходных данных, изображаемых стрелками (см. рисунок 2.60).

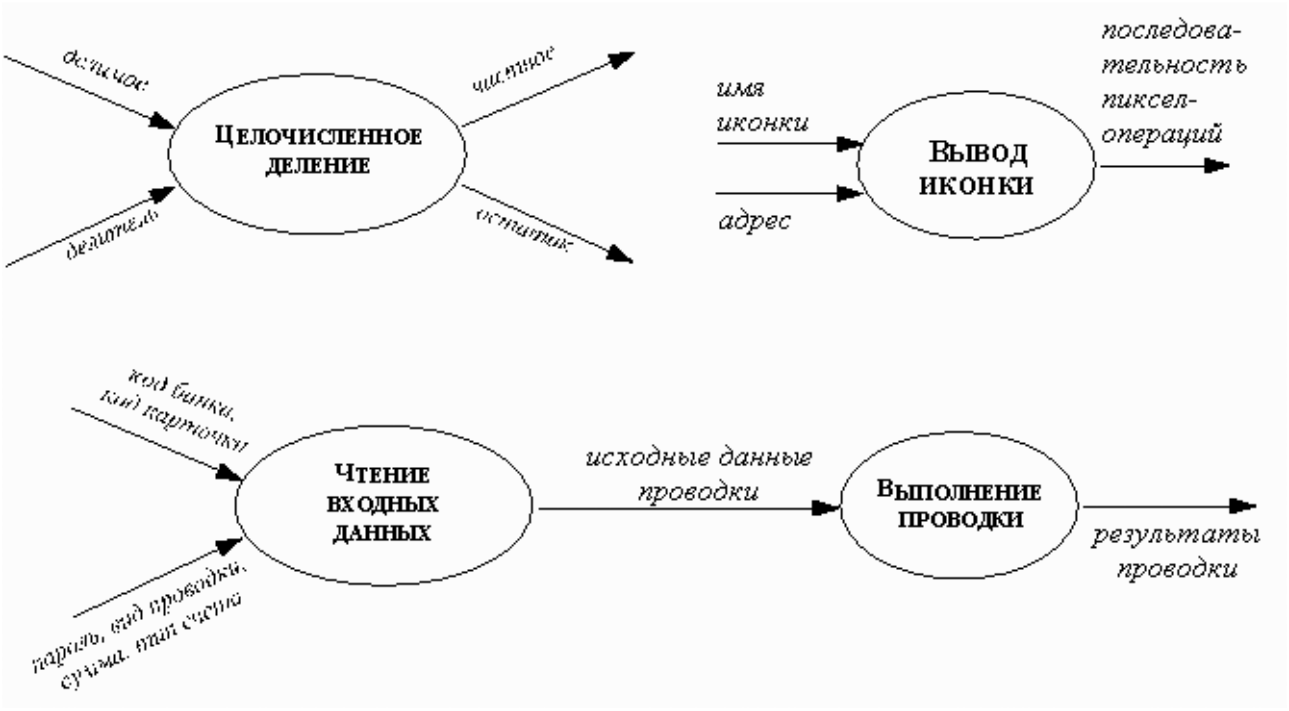

Рис. 2.60. Примеры процессов

Процессы реализуются в виде методов (или их частей) и соответствуют операциям конкретных классов.

Потоки данных

Поток данных соединяет выход объекта (или процесса) со входом другого объекта (или процесса). Он представляет промежуточные данные вычислений. Поток данных изображается в виде стрелки между производителем и потребителем данных, помеченной именами соответствующих данных; примеры стрелок, изображающих потоки данных, представлены на рисунке 2.61. На первом примере изображено копирование данных при передаче одних и тех же значений двум объектам, на втором - расщепление структуры на ее поля при передаче разных полей структуры разным объектам.

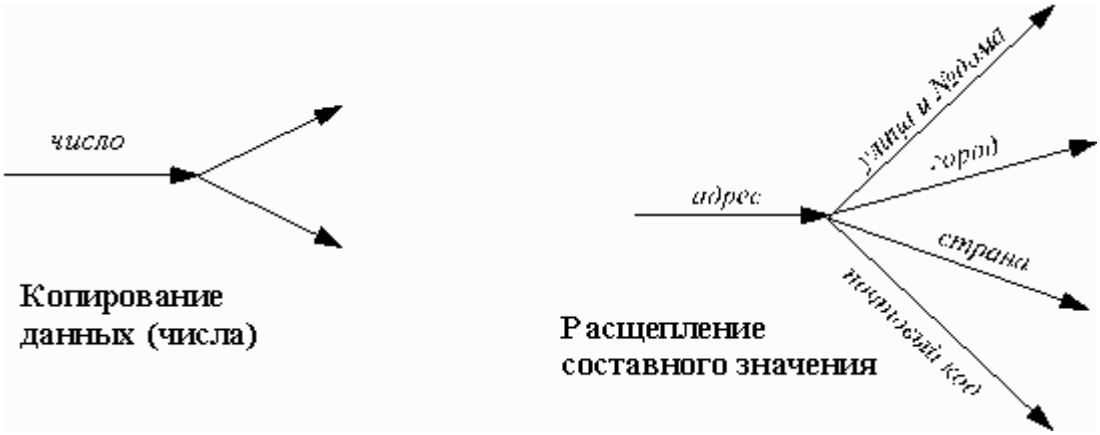

Рис. 2.61. Потоки данных

Активные объекты

Активным называется объект, который обеспечивает движение данных, поставляя или потребляя их. Активные объекты обычно бывают присоединены к входам и выходам ДПД. Примеры активных объектов показаны на рисунке 2.62. На ДПД активные объекты обозначаются прямоугольниками.

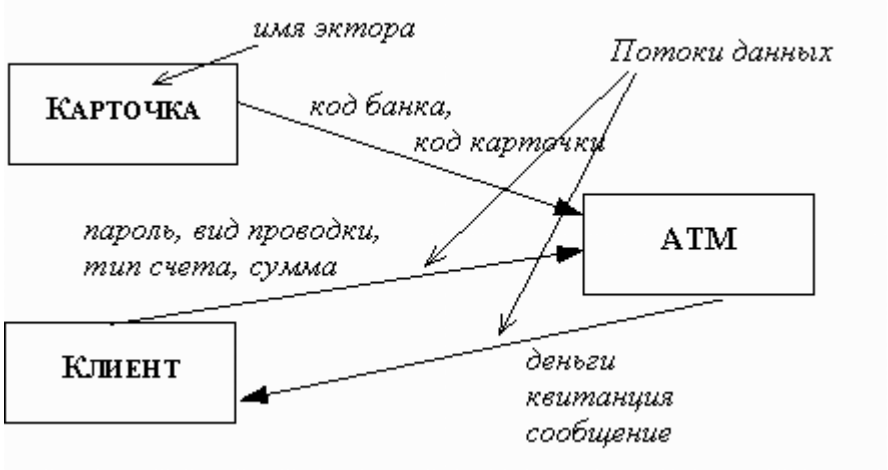

Рис. 2.62. Активные объекты (экторы)

Хранилища данных

Хранилище данных - это пассивный объект в составе ДПД, в котором данные сохраняются для последующего доступа. Хранилище данных допускает доступ к хранимым в нем данным в порядке, отличном от того, в котором они были туда помещены. Агрегатные хранилища данных, как например, списки и таблицы, обеспечивают доступ к данным в порядке их поступления, либо по ключам. Примеры хранилищ данных приведены на рисунке 2.63.

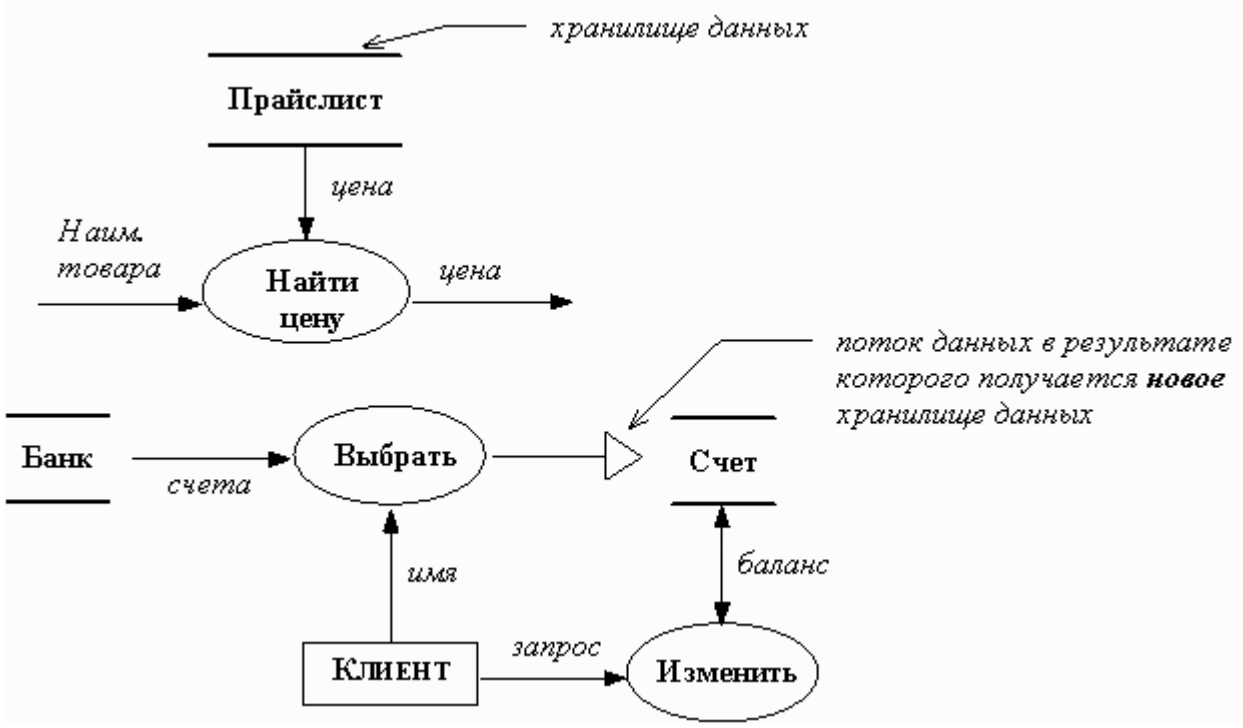

Рис. 2.63. Хранилища данных

Потоки управления

ДПД показывает все пути вычисления значений, но не показывает в каком порядке значения вычисляются. Решения о порядке вычислений связаны с управлением программой, которое отражается в динамической модели. Эти решения, вырабатываемые специальными функциями, или предикатами, определяют, будет ли выполнен тот или иной процесс, но при этом не передают процессу никаких данных, так что их включение в функциональную модель необязательно. Тем не менее иногда бывает полезно включать указанные предикаты в функциональную модель, чтобы в ней были отражены условия выполнения соответствующего процесса. Функция, принимающая решение о запуске процесса, будучи включенной в ДПД, порождает в ДПД поток управления (он изображается пунктирной стрелкой).

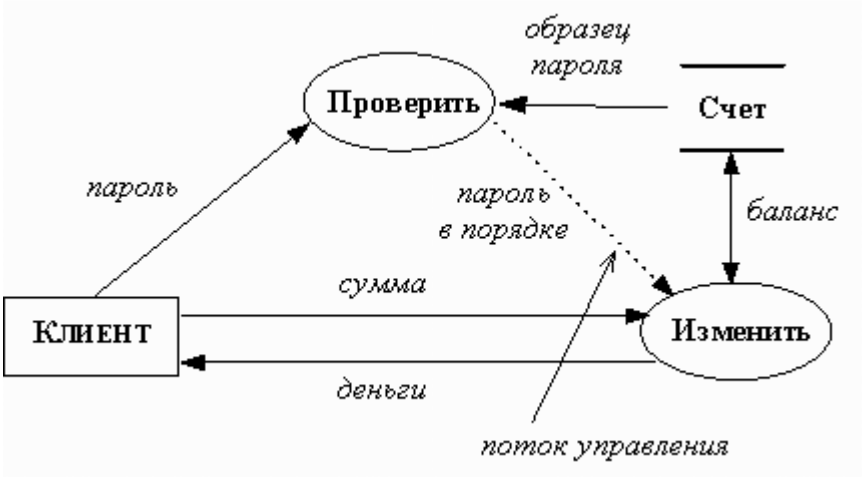

Рис. 2.64. Поток управления

На рисунке 2.64 изображен пример потока управления: клиент, желающий снять часть своих денег со счета в банке, вводит в АТМ пароль и требуемую сумму, однако фактическое снятие и выдача денег происходит только в том случае, когда введенный пароль совпадает с его образцом.

Несмотря на то, что потоки управления иногда оказываются весьма полезными, следует иметь в виду, что включение их в ДПД приводит к дублированию информации, входящей в динамическую модель.

2.6.2. Описание операций

Процессы ДПД в конце концов должны быть реализованы как **операции** объектов. Каждый процесс нижнего (базового) уровня, так же как и процессы верхних уровней, в состав которых входят процессы более нижних уровней, реализуются как операции. При этом реализация процессов верхних уровней может отличаться от их представления на ДПД, так как при реализации обычно производится их оптимизация: в результате оптимизации процессы нижних уровней, составляющие процесс более высокого уровня могут "слиться", после чего они станут невидимы.

Все операции должны быть специфицированы. Спецификация операции содержит ее сигнатуру (имя операции, количество, порядок и типы ее параметров, количество, порядок и типы возвращаемых ею значений) и описание ее эффекта (действия, преобразования). Для описания эффекта операции можно использовать:

- ✓ математические формулы;
- ✓ табличные функции: таблицы, сопоставляющие выходные значения входным;
- ✓ уравнения, связывающие входные и выходные значения;

- ✓ аксиоматическое определение операций с помощью пред- и пост-условий;
- ✓ таблицы принятия решений;
- ✓ псевдокод;
- ✓ естественный язык.

Пример описания операции (эффект ее описан на естественном языке) приведен на рисунке 2.65.

```
изменить_счет (счет, сумма, вид_проводки) -> деньги, квитанция
  если сумма снимается и больше баланса счета,
    то "отменить_проводку"
  если сумма снимается и меньше баланса счета,
    то "дебетовать_счет" и "выдать_деньги"
  если сумма вносится на счет
    то "кредитовать_счет"
  если запрос
    то "выдать_запрос"
  во всех случаях:
    квитанция должна содержать номер АТМ, дату, время,
      номер счета, вид проводки, сумму проводки (если она
      есть), новый баланс счета
```

Рис. 2.65. Спецификация операции изменить_счет (при описании эффекта операции использованы операции отменить_проводку, выдать_запрос, выдать_деньги, дебетовать_счет и кредитовать_счет)

Внешняя спецификация операции описывает только те изменения, которые видны вне операции. Операция может быть реализована таким образом, что при ее выполнении будут использоваться некоторые значения, определенные внутри операции (например, в целях оптимизации), некоторые из этих значений могут даже быть частью состояния объекта. Эти детали реализации операции скрыты от остальных объектов и не участвуют в определении внешнего эффекта операции. Изменения внутреннего состояния объекта, не видные вне его, не меняют значения объекта.

Все нетривиальные операции можно разделить на три категории: запросы, действия и активности. Запросом называется операция без побочных эффектов над видимым извне объекта его состоянием (чистая функция). Запрос, у которого нет параметров, кроме целевого объекта, является производным атрибутом. Например, для точки на координатной плоскости, радиус и полярный угол - производные атрибуты; из этого примера видно, что между основными и производными атрибутами нет принципиальной разницы, и выбор основных атрибутов во многом случаен.

Действием называется операция, имеющая побочные эффекты, которые могут влиять на целевой объект и на другие объекты системы, которые достижимы из целевого объекта. Действие не занимает времени (логически, оно совершается мгновенно). Каждое действие может быть определено через те изменения, которые оно производит в состоянии объекта, меняя значения его атрибутов и связей; в каком порядке производятся эти изменения, несущественно: мы считаем, что все они происходят одновременно и мгновенно. Наиболее распространенным способом описания действия является задание алгоритма его выполнения на компьютере.

Активностью называется операция, выполняемая объектом, или над объектом, выполнение которой занимает определенное время. Активность имеет побочные эффекты. Активности

могут быть только у активных объектов, так как пассивные объекты есть попросту склады данных.

2.6.3. Ограничения

Ограничение указывает на зависимость между соответственными значениями двух объектов, либо между различными значениями одного объекта. Ограничение может быть выражено в виде некоторой функции (количественное ограничение), либо отношения (качественное ограничение). Нас интересуют ограничения на атрибуты объектов, а также на состояния и события. Важным видом ограничений являются инварианты: утверждения о том, что значение некоторой функции от атрибутов, состояний и событий остается постоянным при функционировании объекта.

2.6.4. Функциональная модель банковской сети

Функциональная модель показывает как вычисляются значения в системе и как они зависят одно от другого. Для конструирования функциональной модели необходимо выполнить следующее:

- определить входные и выходные значения;
- построить ДПД, показывающие функциональные зависимости;
- описать функции;
- описать ограничения;
- сформулировать критерии оптимизации;
- уточнить набор операций в объектной модели.

Выполним все эти шаги, чтобы построить функциональную модель банковской сети (АТМ).

Определение входных и выходных значений

Начнем с определения входных и выходных значений. Эти значения являются параметрами событий между системой и окружающим ее миром. Входные и выходные значения банковской сети показаны на рисунке 2.66 (пунктирная линия показывает границу системы).

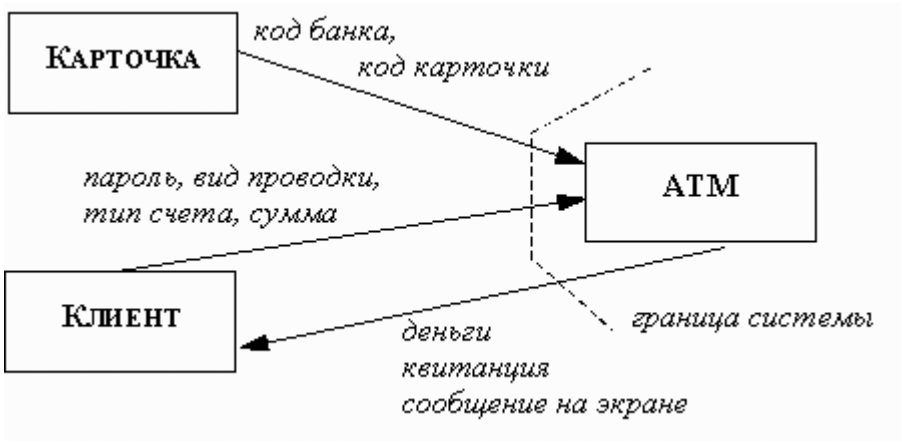

Рис. 2.66. Входные и выходные значения банковской сети

Поскольку все взаимодействия между внешним миром и системой проходят через АТМ (или кассовый терминал, который здесь не рассматривается), все входные и выходные значения являются параметрами событий, связанных с объектом АТМ (банкомат).

Построение ДПД

ДПД обычно строится по уровням. На верхнем уровне, как правило, показывают один единственный процесс, или, как в нашем примере (рисунок 2.67), процесс ввода, основной процесс вычисления требуемых значений и процесс вывода. Входные и выходные данные на этой диаграмме поставляются и потребляются внешними объектами (клиент, карточка).

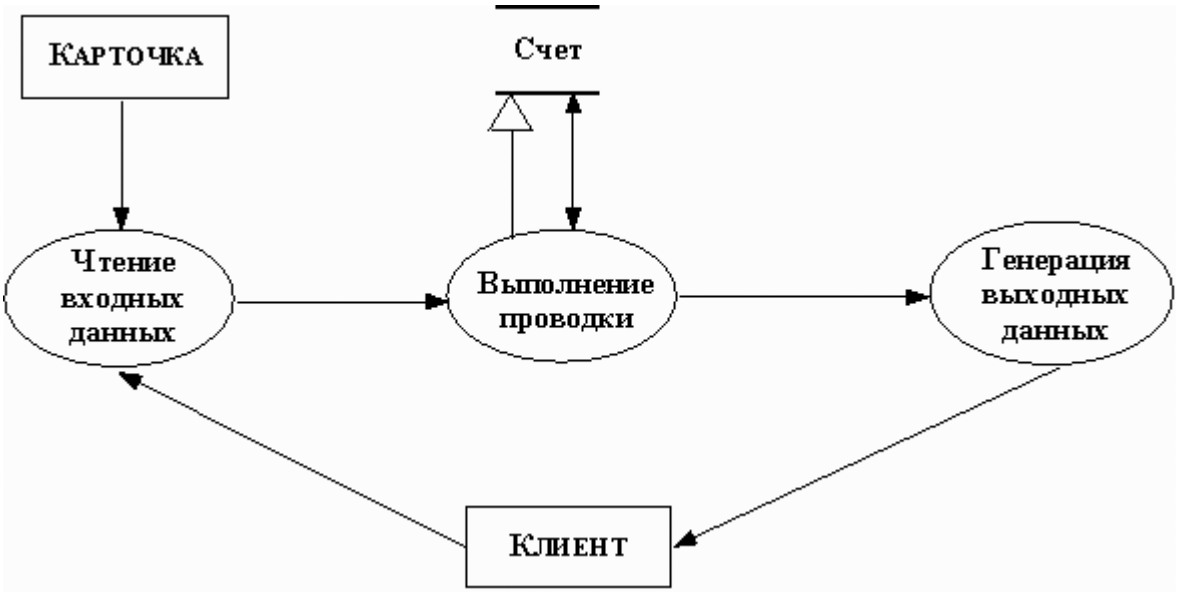

Рис. 2.67. Процессы верхнего уровня в системе обслуживания банковской сети

На ДПД каждого уровня указываются элементы данных более высоких уровней, чтобы показать данные, обрабатываемые каждым процессом (см. рисунок 2.68). Большая часть систем содержит внутренние хранилища данных, в которых хранятся данные в периодах между взаимодействиями. Для процесса выполнить проводку таким хранилищем является объект счет. Внутренние хранилища данных отличаются тем, что получаемые ими данные не выдаются немедленно, а хранятся в объекте для использования в будущем.

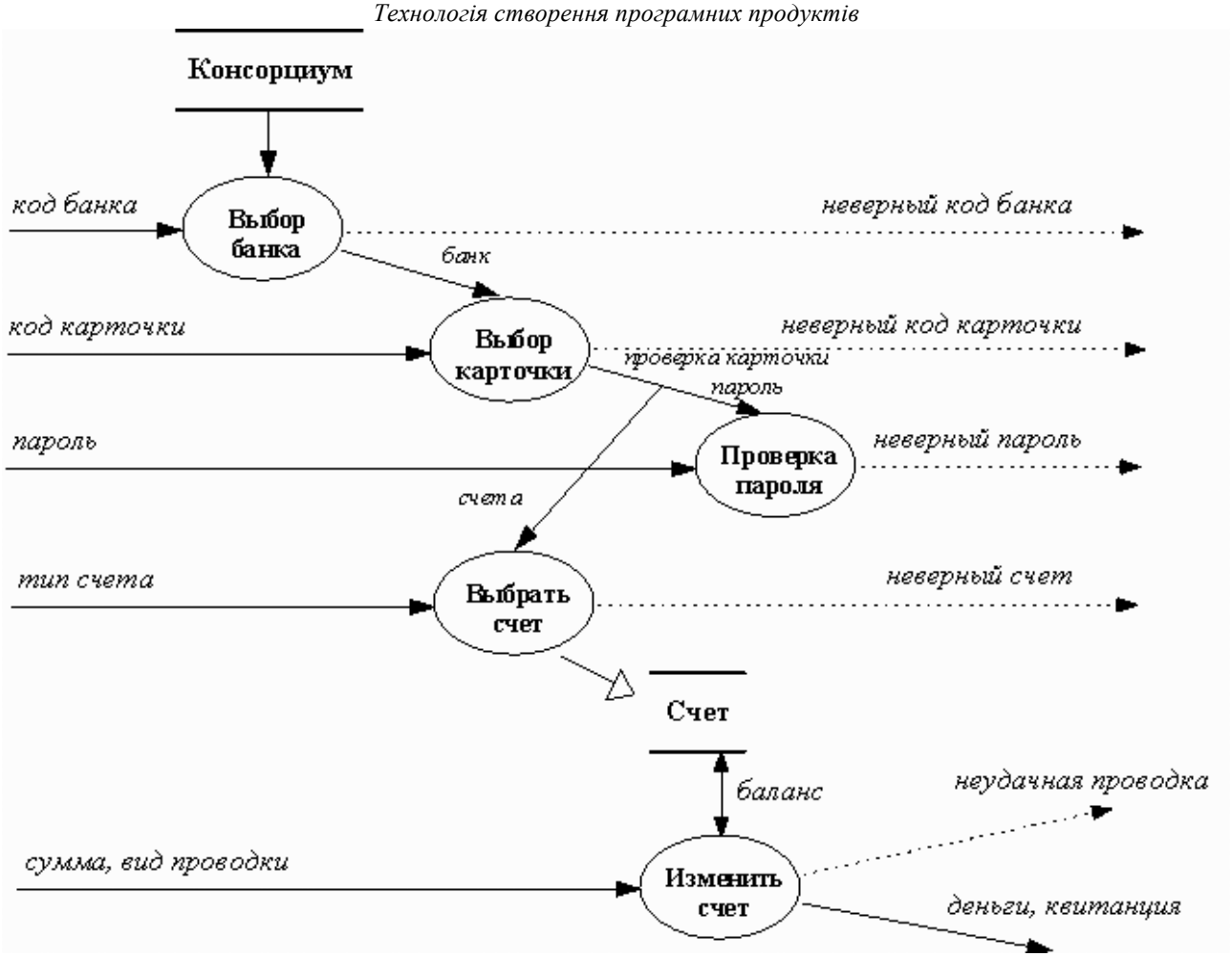

Рис. 2.68. ДПД процесса выполнить проводку в системе обслуживания банковской сети

ДПД показывают только зависимости между операциями; они не содержат информации о последовательности выполнения операций (операции, показанные на ДПД могут быть не всегда выполняемыми или исключающими одна другую). Например, пароль должен быть подтвержден до внесения изменений в счет; если он указан неверно, изменения в счет не вносятся. Последовательность выполнения операций указывается в динамической, а не в функциональной модели.

Некоторые значения данных влияют на выбор последовательности вычислений в динамической модели; тем не менее эти значения не влияют непосредственно на выходные значения ДПД, так как ДПД показывает все возможные пути вычислений. Однако может оказаться полезным поместить функции, осуществляющие выбор последовательности вычислений, и в функциональную модель, так как эти функции могут выражать сложные зависимости от входных данных. Такие функции показаны на ДПД, но их выходными значениями являются управляющие сигналы, показанные пунктирными линиями.
Примером такой функции является функция проверить пароль.

Описание функций

После того как ДПД составлена, необходимо составить описание каждой функции из ДПД. Пример описания функции приведен на рисунке 2.65. Мы не приводим других примеров описания функций банковской сети в виду их тривиальности.

Описание ограничений

Ограничения - это функциональные зависимости между объектами, не сводящиеся к непосредственным зависимостям между выходами и входами. Ограничения могут накладываться одновременно на два объекта, на различные объекты одного класса в различные моменты времени (инвариант), или на объекты разных классов в различные моменты времени (хотя в последнем случае это, как правило, связи между входами и выходами). Ограничения, которым должны удовлетворять входные значения функции, называются ее предусловиями, а ограничения, которым удовлетворяют выходные значения функции, - ее постусловиями.

В банковской сети можно ввести следующие ограничения:
"счет не может иметь отрицательного баланса";
"отрицательный баланс кредитуемого счета не может быть больше лимита кредитования для этого счета".

Определение критериев оптимизации

Определяем какие значения необходимо максимизировать или минимизировать в процессе работы системы; если критериев оптимизации несколько и они противоречат один другому, следует определить компромиссные решения. На этой стадии не следует принимать по этому поводу слишком точных решений, так как в процессе дальнейшего проектирования критерии будут уточняться и изменяться.

Оптимизационные критерии для банковской сети:
минимизировать количество физических пересылок между различными узлами сети;
минимизировать время, в течение которого счет закрыт для доступа.

Операции вводятся на различных стадиях разработки модели проектируемой системы (см. выше):

- при разработке объектной модели;
- при определении событий;
- при определении действий и активностей;
- при определении функций.

На заключительном этапе разработки модели проектируемой системы все введенные операции вводятся в ее объектную модель.

К перечисленным операциям добавляются операции, связанные с внешним поведением объектов рассматриваемых классов: эти операции определяются природой объектов, а не свойствами разрабатываемой системы. Такие операции расширяют определение объекта, выводя его за рамки непосредственных требований со стороны конкретной системы. Примерами таких операций для банковской сети являются: закрыть счет, создать сберегательный счет, создать банковскую карточку и т.п. С точки зрения системы эти операции не имеют смысла, но они отражают реальные свойства классов счет и карточка и могут оказаться полезными при использовании этих классов в других системах. В заключение следует исследовать объектную модель на предмет упрощения операций ее классов. Пример банковской сети недостаточно сложен, чтобы продемонстрировать это.

2.7. Заключительные замечания к разделу

Цель анализа - обеспечить правильную постановку и адекватное понимание рассматриваемой прикладной задачи, помочь убедиться, что предварительно спроектированная прикладная система сможет удовлетворить заказчика. Хороший анализ охватывает все существенные особенности задачи, не внося каких-либо реализационных

особенностей в предварительный проект системы. Тем самым обеспечивается свобода реализационных решений на этапе реализации.

Объектная модель показывает статическую структуру проблемной области, для которой разрабатывается система. Сначала определяются классы объектов, затем зависимости между объектами, включая агрегацию. Для упрощения структуры классов используется наследование. Объектная модель должна содержать краткие комментарии на естественном языке.

Динамическая модель показывает поведение системы, в особенности последовательность взаимодействий. Сначала готовятся сценарии типичных сеансов взаимодействия с системой, затем определяются внешние события, отражающие взаимодействие системы с внешним миром; после этого строится диаграмма состояний для каждого активного объекта, на которой представлены образцы событий, получаемых системой и порождаемых ею, а также действий, выполняемых системой. Построенные диаграммы состояний сравниваются между собой, чтобы убедиться в их непротиворечивости. На этом построение динамической модели заканчивается.

Функциональная модель показывает функциональный вывод значений безотносительно к тому, когда они вычисляются. Сначала определяются входные и выходные значения системы как параметры внешних событий. Затем строятся диаграммы потоков данных, показывающие как вычисляется каждое выходное значение по входным и промежуточным значениям. Диаграммы потоков данных выявляют взаимодействие с внутренними объектами системы, которые служат хранилищами данных в периоды между сеансами работы системы. В заключение определяются ограничения и критерии оптимизации.