Учебный курс

Язык UML в анализе и проектировании программных систем и бизнес-процессов

Лекция 4

Диаграмма последовательности языка UML 2

Автор:

Леоненков Александр Васильевич

кандидат технических наук,
старший научный сотрудник

Диаграмма последовательности (sequence diagram)

- – диаграмма, которая служит для представления взаимодействия элементов модели в форме последовательности сообщений и соответствующих событий на линиях жизни объектов
- Масштаб для оси времени на диаграмме последовательности не указывается, поскольку эта диаграмма предназначена для моделирования только лишь временного порядка следования сообщений типа "раньше-позже”
- *Взаимодействие (interaction)* — единица поведения некоторого классификатора, которая концентрирует внимание на наблюдаемом обмене информацией между элементами, являющимися участниками этого взаимодействия

Графическая нотация представления взаимодействия

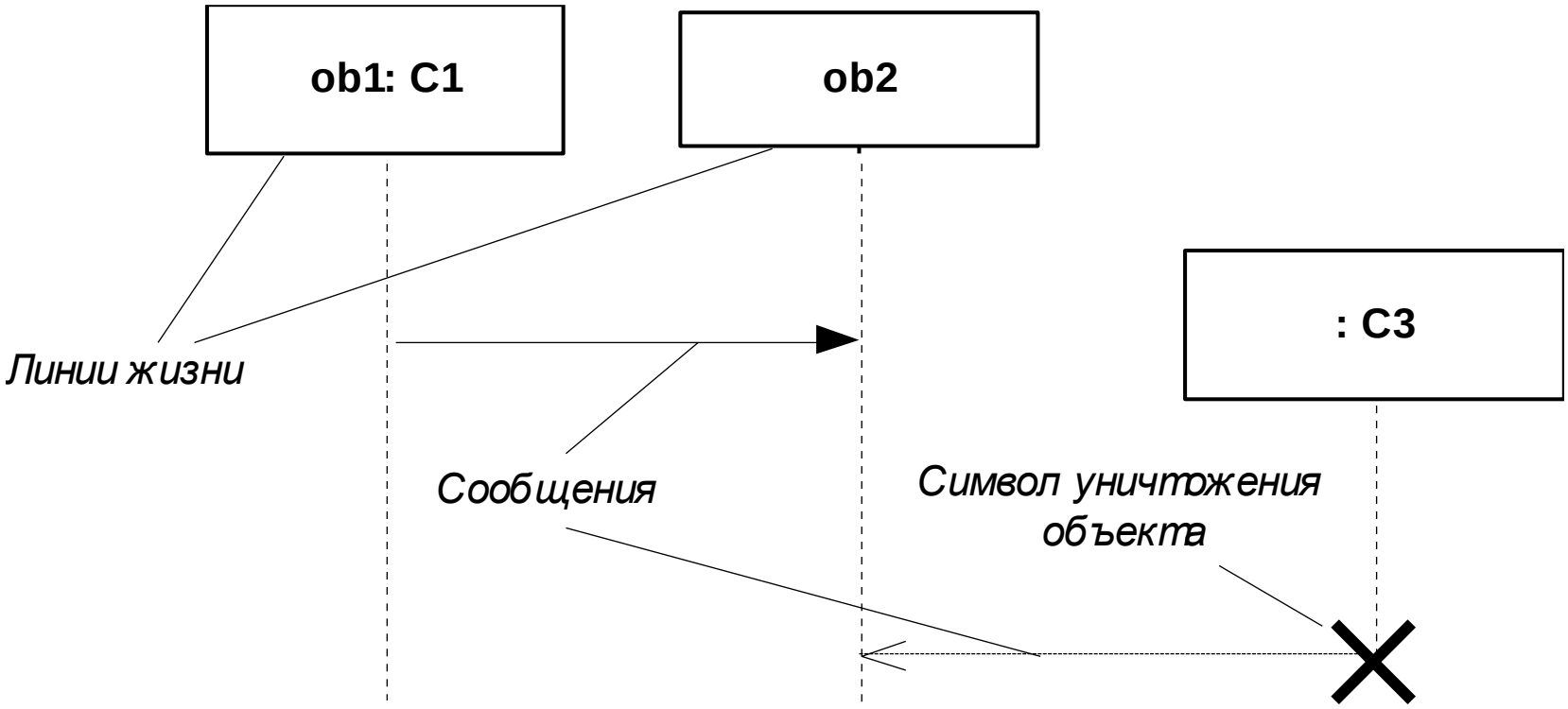
- - прямоугольник с непрерывными сторонами, который также называется *фреймом* (frame) диаграммы
- В верхнем левом углу прямоугольника фрейма изображается небольшой пятиугольник, в который помещается ключевое слово **sd**, за которым следует имя взаимодействия и его параметры
- Порядок наступления событий вдоль линий жизни имеет значение для обозначения последовательности, в которой эти наступления события происходят
- Однако, абсолютные расстояния между наступлениями событий на линиях жизни не имеют семантики
- Другими словами, время на диаграмме последовательности имеет шкалу порядка, а не шкалу отношений, о чем необходимо знать всем разработчикам

Линия жизни *(lifeline)*

- - представляет одного индивидуального участника взаимодействия или отдельную взаимодействующую сущность
- Информация, идентифицирующая линию жизни, изображается внутри прямоугольника в следующем формате (БНФ):

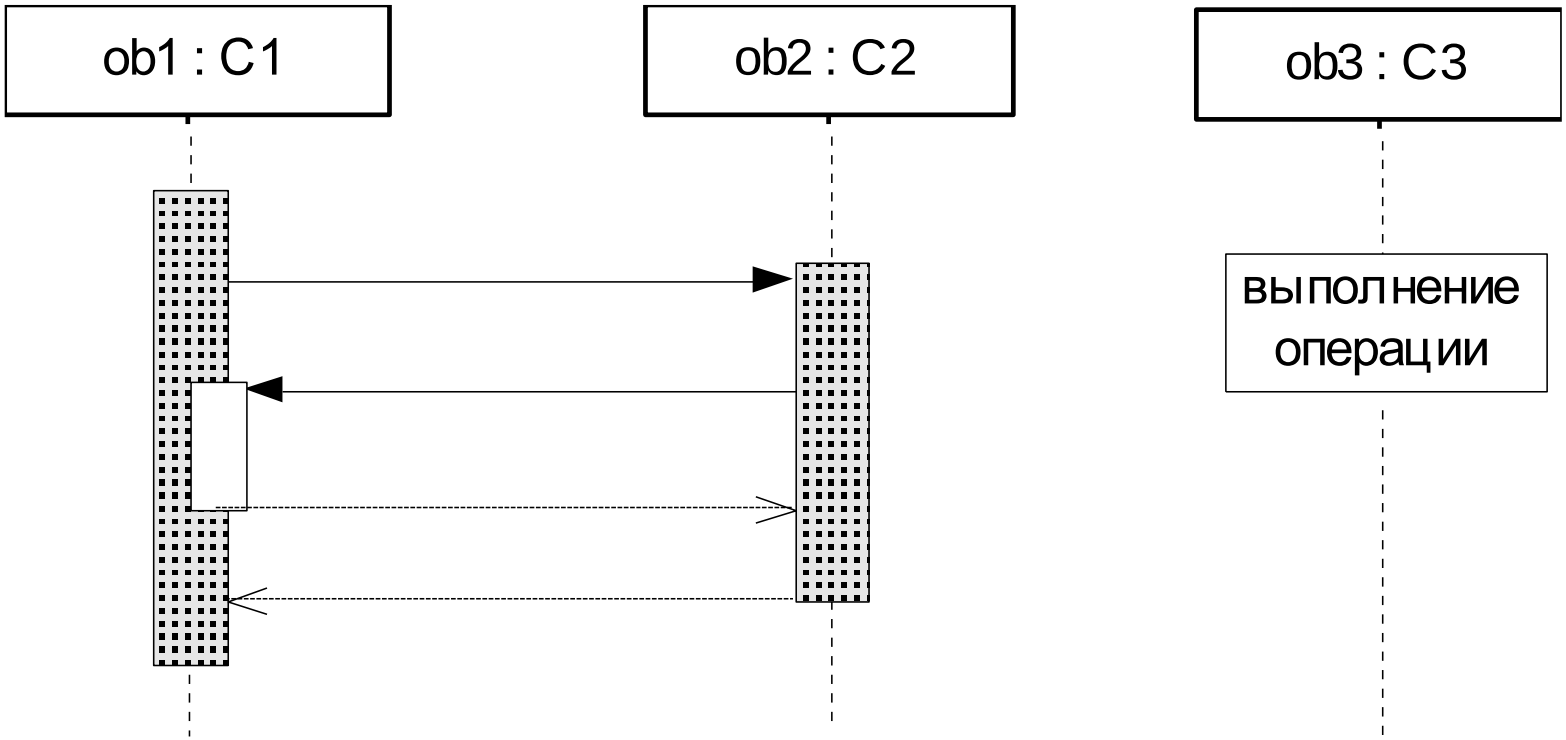
  <идентификатор-линии-жизни>::= ([<имя-связанного-элемента> [‘[‘<селектор>‘] ’]] [: <имя-класса>] [декомпозиция]) | ‘self’,
- *где <селектор>::= <выражение>*
- *<декомпозиция>::=‘**ref**’ <идентификатор-взаимодействия> [‘**strict**’].*
- Здесь <*имя-класса*>является типом, на который ссылается представленный соединяемый элемент. Если именем является ключевое слово **self**, то такая линия жизни представляет объект классификатора, который владеет данным взаимодействием

Нотация линии жизни



Спецификация выполнения *(execution specification)*

- - предназначена для моделирования состояния активности линии жизни в описываемом взаимодействии.



Сообщение *(message)*

- – элемент модели, предназначенный для представления отдельной коммуникации между линиями жизни некоторого взаимодействия
- Имя сообщения имеет следующий синтаксис:

  <идентификатор-сообщения>::= ([<атрибут>‘=’] <имя-операции-или-сигнала> [‘(‘ [<аргумент> [‘,’<аргумент>] ‘)’] [‘:’ <возвращаемое-значение>])|‘*’,*
- *где <аргумент> ::= (<[имя-параметра>‘=’] <значение-аргумента>) | (<атрибут> ‘=’ <имя-out-параметра> [‘:’ <значение-аргумента>]|‘ -’*

Сорт сообщения *(message sort)*

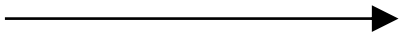
- – представляет собой тип перечисления, который идентифицирует характер коммуникации, которая лежит в основе генерации данного сообщения
- *synchCall' – синхронное* сообщение, которое соответствует синхронному вызову операции
- Синхронные сообщения обычно представляют вызовы методов и изображаются сплошной линией с закрашенной стрелкой

Сорт сообщения

- *asynchCall’* – *асинхронное* сообщение, которое соответствует асинхронному вызову операции, изображаются сплошной линией с открытой стрелкой в форме буквы “V”.
- *asynchSignal’* – *асинхронный* сигнал, которое соответствует некоторому асинхронному действию, изображаются сплошной линией с открытой стрелкой в форме буквы.
- *ответное* (reply) от вызова метода, изображается пунктирной линией с открытой стрелкой в форме буквы “V”
- Сообщение *создания объекта* (object creation) также изображается пунктирной линией с открытой стрелкой в форме буквы “V”

Вид сообщения *(message kind)*

- *complete – полное* сообщение, для которого существует событие передачи и событие приема, изображаются рассмотренным ранее образом в зависимости от сорта сообщения.
- *unknown – неизвестное* сообщение, для которого отсутствуют событие передачи и событие приема. Эти сообщения не должны представляться на диаграмме последовательности.



Вид сообщения

- *lost – потерянное* сообщение, для которого существует событие передачи и отсутствует событие приема, изображается в форме небольшого черного круга на конце стрелки сообщения. Оно интерпретируется как сообщение, которое никогда не достигнет своего места назначения
- *found – найденное* сообщение, для которого существует событие приема и отсутствует событие передачи, изображается в форме небольшого черного круга на начальном конце сообщения. Оно интерпретируется как сообщение, инициатор которого находится за пределами области описания

Сигнал *(signal)*

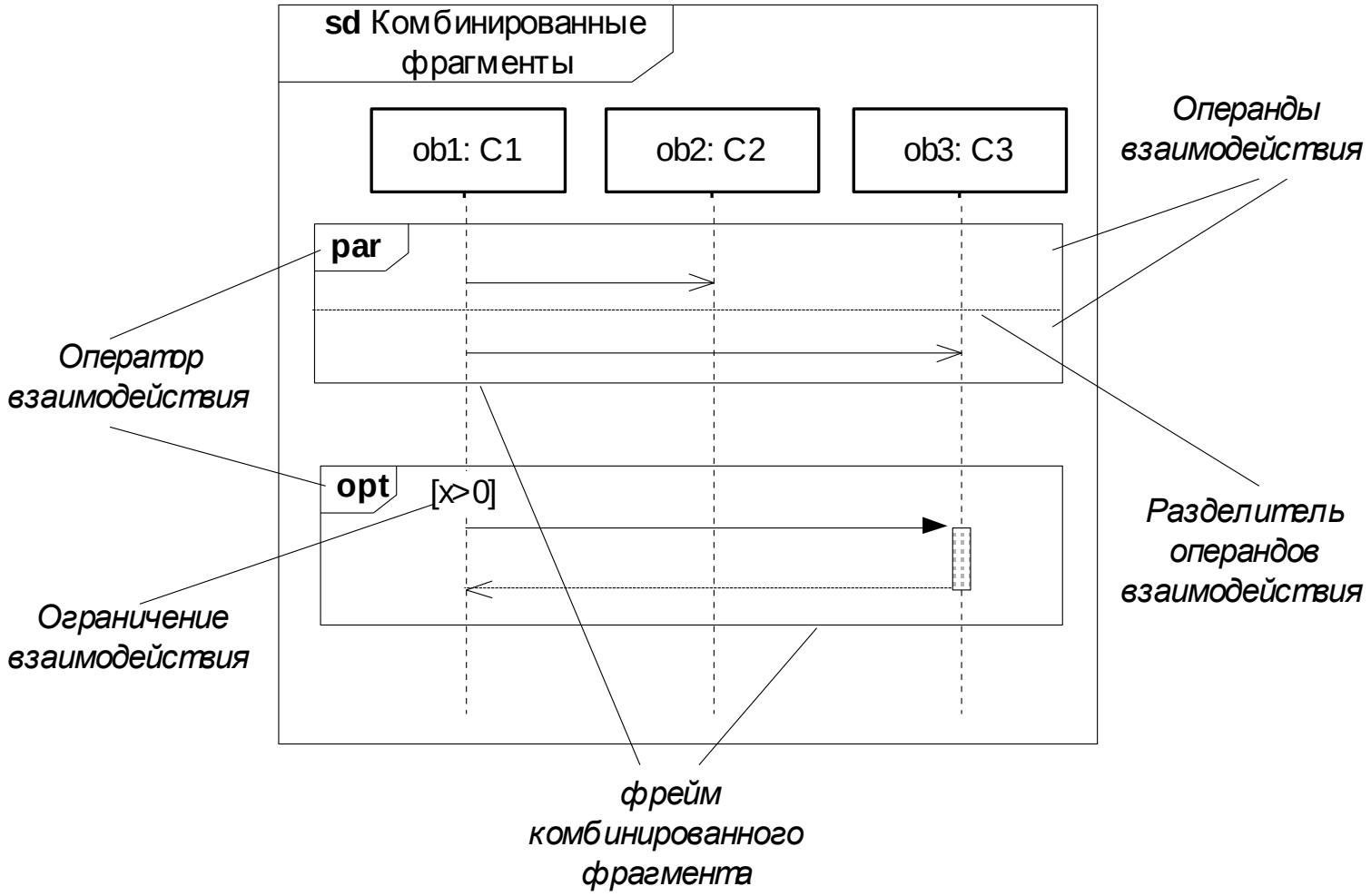
- - представляет собой спецификацию асинхронной коммуникации между линиями жизни
- *Событие сигнала (signal event)* представляет собой прием линией жизни некоторого асинхронного сигнала
- Спецификация события сигнала обозначается с использованием следующего формата (БНФ) :

  <событие-сигнала>::= <имя-сигнала> [‘(‘ [<спецификация-назначения>] ‘)’],
- где *<спецификация-назначения>::= <имя-атрибута> [‘,’<имя-атрибута>]*.*

Комбинированный фрагмент *(combined fragment)*

- – элемент модели, предназначенный для представления внутренней логической структуры фрагментов взаимодействия
- *Операнд взаимодействия (interaction operand)* – отдельный фрагмент взаимодействия, предназначенный для использования в качестве внутренней части комбинированного фрагмента
- *Ограничение взаимодействия (interaction constraint)* представляет собой логическое выражение, которое выступает в роли сторожевого условия некоторого операнда в комбинированном фрагменте

Графическое изображение комбинированного фрагмента



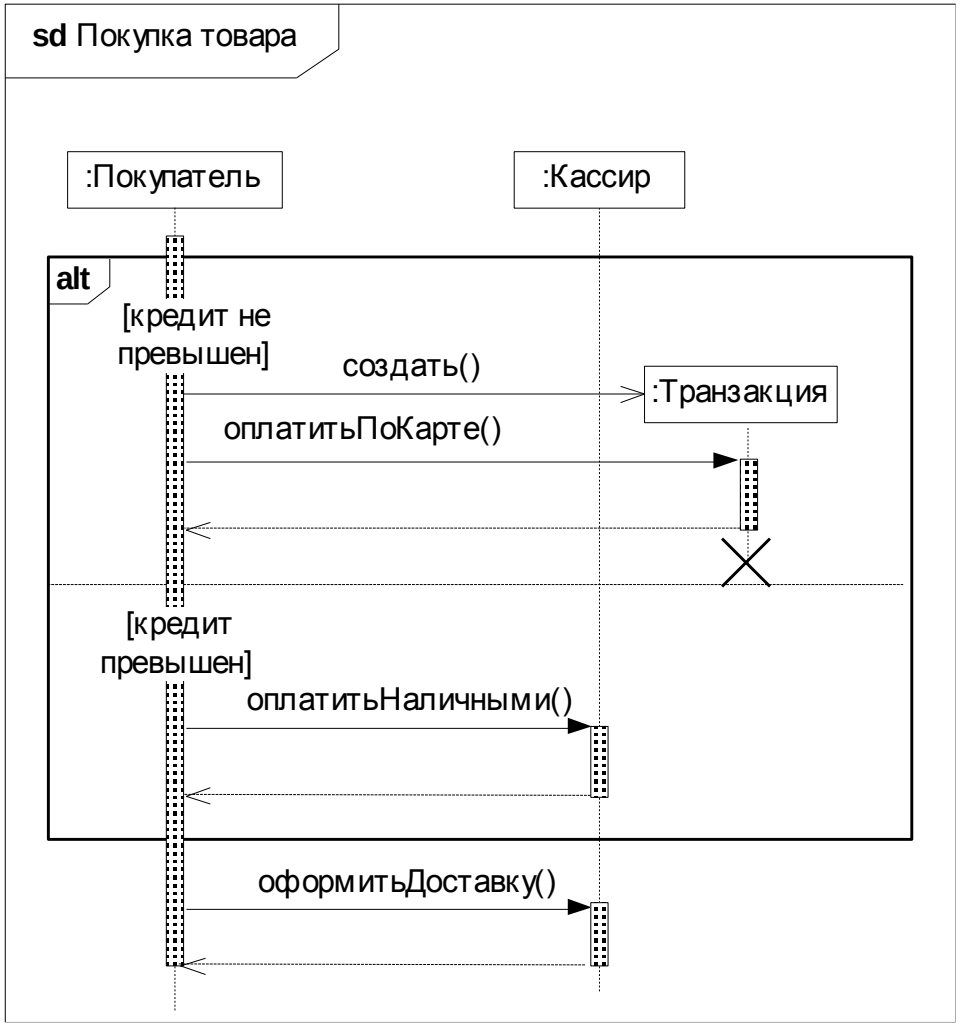
Оператор взаимодействия (interaction operator)

- - определяет тип комбинированного фрагмента и является перечислением следующих 12 литералов:
  - **alt** **assert**
  - **break** **critical**
  - **ignore** **consider**
  - **loop** **neg**
  - **opt** **par**
  - **seq** **strict**

1. Альтернативы (alt)

- Оператор взаимодействия **alt** специфицирует комбинированный фрагмент *Альтернативы* (alternatives), который представляет некоторый выбор поведения
- Выбор может быть сделан не более одного из операндов
- Выбранный операнд должен иметь явное или неявное выражение сторожевого условия, которое в этой точке взаимодействия должно принимать значение «истина»
- Если операнд не имеет никакого сторожевого условия, то неявно предполагается, что сторожевое условие имеет значение «истинна»
- Операнд, помеченный сторожевым условием [else], обозначает отрицание дизъюнкции всех других сторожевых условий этого комбинированного фрагмента

Пример комбинированного фрагмента Альтернативы



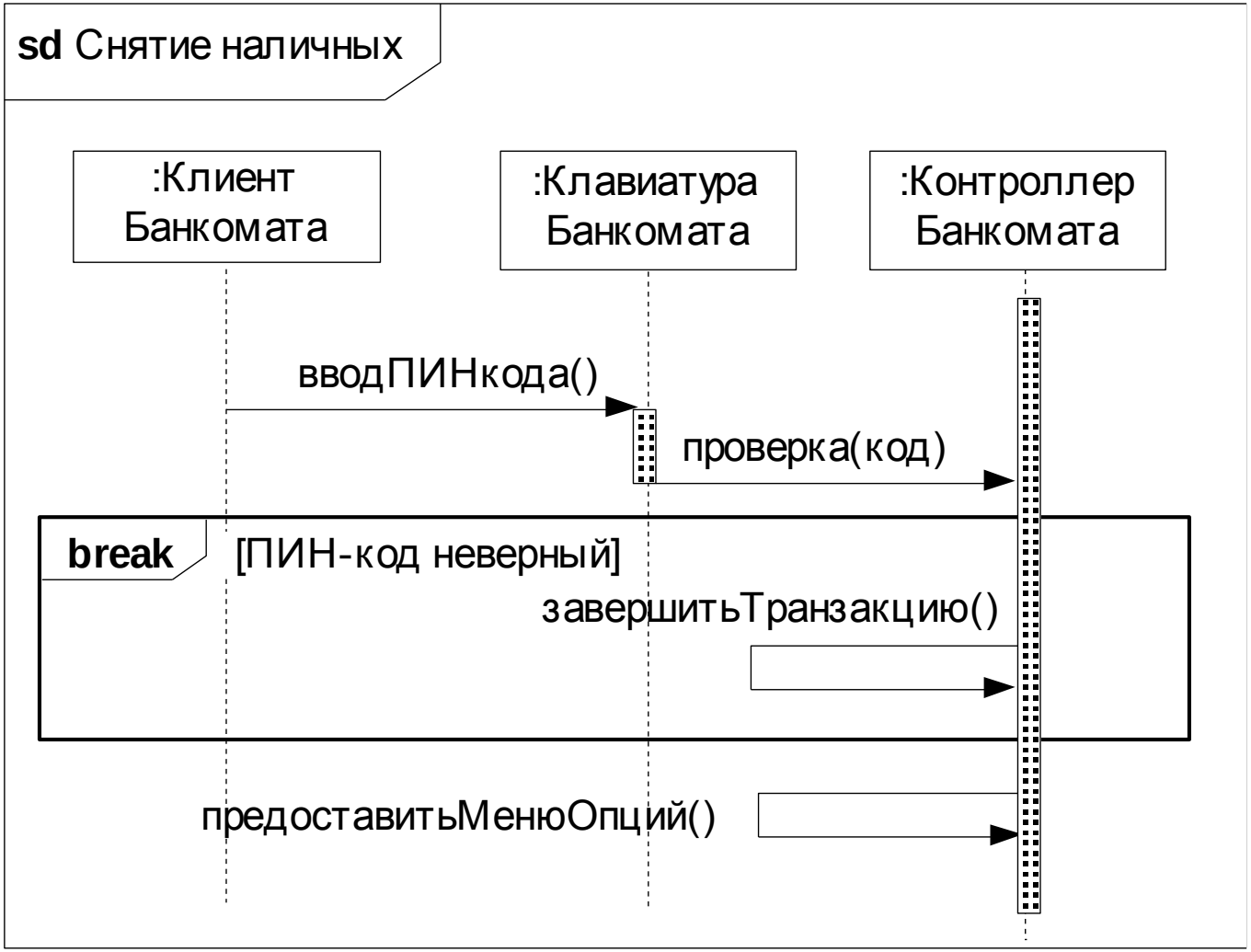
2. Утверждение (assert)

- Оператор взаимодействия **assert** специфицирует комбинированный фрагмент *Утверждение* (assertion), который представляет некоторое утверждение
- Единственными следствиями, которые имеют возможность продолжения, являются сообщения или вложенные фреймы данного операнда
- Все другие продолжения приводят в результате к недействительным траекториям
- Комбинированный фрагмент Утверждение часто объединяется с операторами **ignore** или **consider**

3. Завершение (break)

- Оператор взаимодействия **break** специфицирует комбинированный фрагмент *Завершение* (break), который представляет некоторый сценарий завершения
- Этот сценарий выполняется вместо оставшейся части фрагмента взаимодействия, который содержит этот соответствующий операнд.
- Обычно оператор Завершение содержит некоторое сторожевое условие
- Если это сторожевое условие принимает значение “истина”, то выполняется комбинированный фрагмент Завершение, а оставшаяся часть фрагмента взаимодействия, содержащего этот операнд, игнорируется
- Если сторожевое условие операнда Завершение принимает значение “ложь”, то операнд Завершение игнорируется, и выполняется оставшаяся часть фрагмента взаимодействия, содержащего этот операнд.

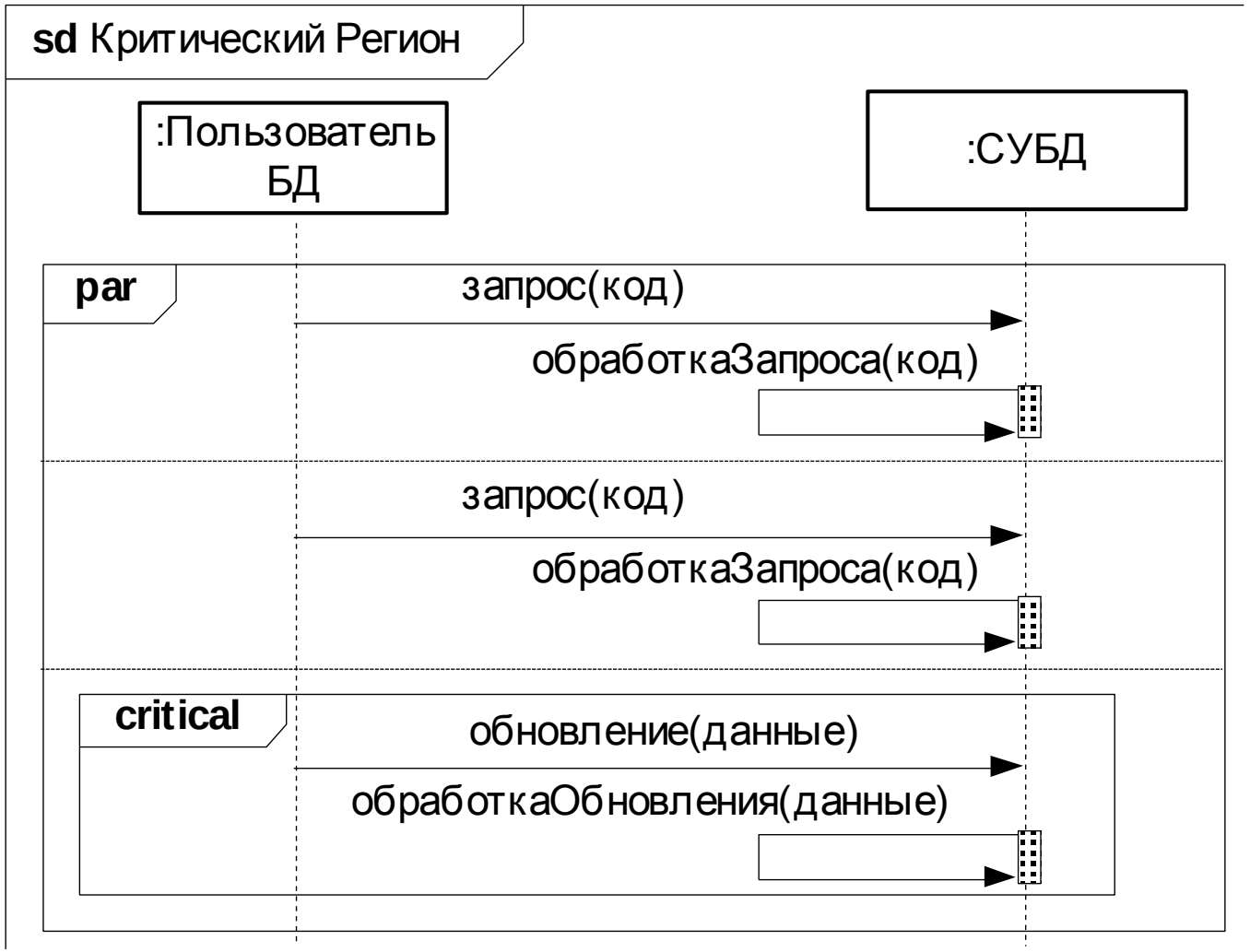
Пример комбинированного фрагмента Завершение



4. Критический регион (critical)

- Оператор взаимодействия **critical** специфицирует комбинированный фрагмент *Критический регион* (critical region), траектории которого не могут чередоваться с другими спецификациями наступления событий на тех линиях жизни, которые этот регион покрывает.
- Критический регион рассматривается как неделимый при определении множества возможных траекторий диаграммы или региона, который его содержит
- Множество траекторий критического региона не может прерываться другими событиями, происходящими вне этого региона
- На практике Критический регион используется, как правило, совместно с оператором параллельности

Пример комбинированного фрагмента Критический регион



5. Рассмотрение(consider)

- Оператор взаимодействия **consider** специфицирует комбинированный фрагмент *Рассмотрение* (consider), в котором изображены только те типы сообщений, какие должны рассматриваться в этом фрагменте
- Это эквивалентно определению того, что при рассмотрении данного фрагмента игнорируются любые другие сообщения, которые не изображены в этом фрагменте.
- Для фрагмента Рассмотрение используется нотация фрейма с оператором, в качестве которого используется ключевое слово **consider**
- Список сообщений следует за операндом и заключается в фигурные скобки согласно следующему формату:

  <*оператор-рассмотрение*>::=‘*consider*“{‘<*имя-сообщения*>[‘,’<*имя-сообщения*>]*‘}’

6. Игнорирование (ignore)

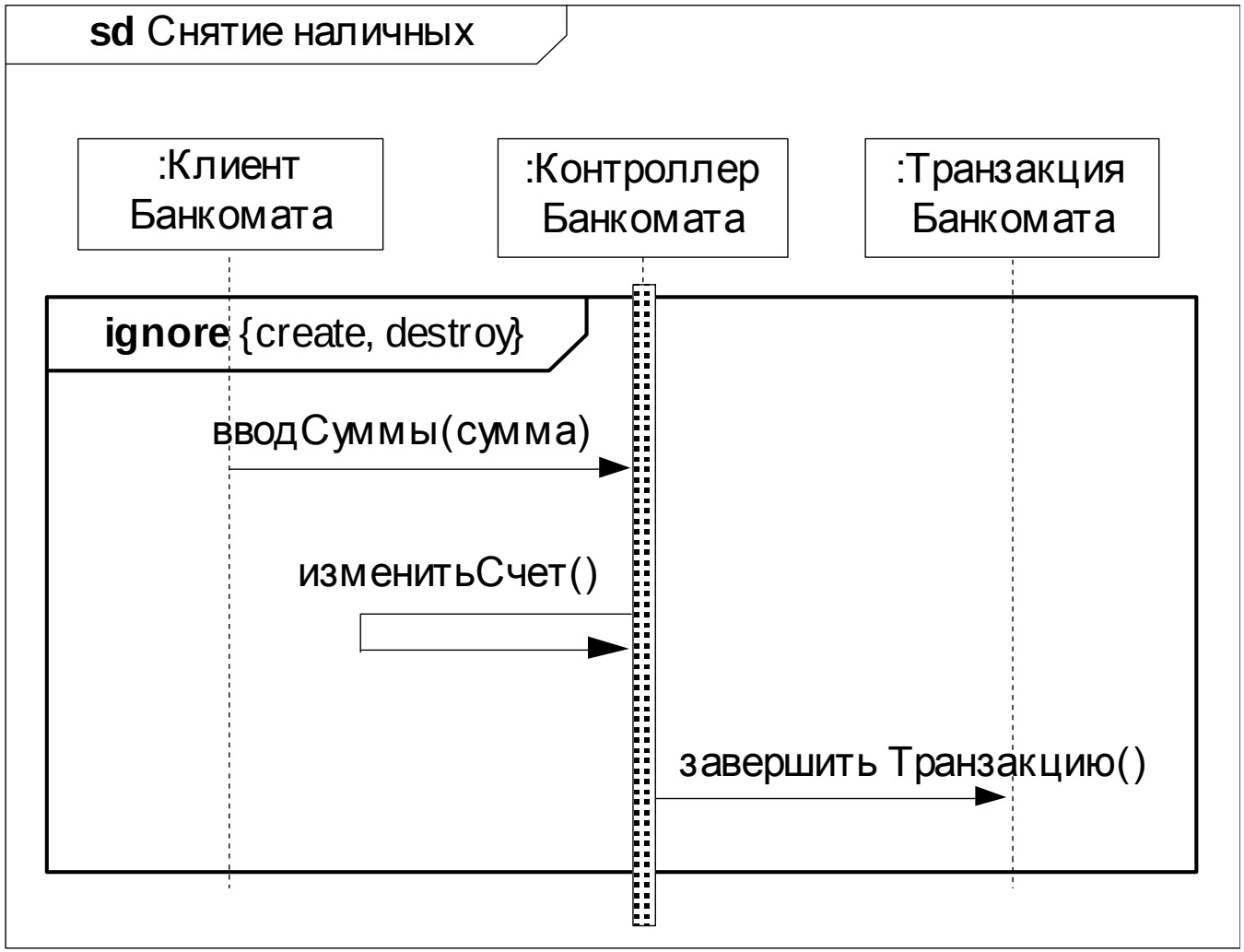
- Оператор взаимодействия **ignore** специфицирует комбинированный фрагмент *Игнорирование* (ignore), в котором имеются некоторые типы сообщений, не изображенные на данной диаграмме
- Эти типы сообщений рассматриваются как несущественные и могут появляться в траекториях при выполнении соответствующего фрагмента.
- Для фрагмента Игнорирование используется нотация фрейма с оператором, в качестве которого используется ключевое слово **ignore**
- Список сообщений следует за операндом и заключается в фигурные скобки согласно следующему формату:

  <*оператор-игнорирование*>::=‘*ignore*”{‘<*имя-сообщения*>[‘,’<*имя-сообщения*>]*‘}’

Примеры Рассмотрение и Игнорирование

- Например, выражение **consider** {m, s} указывает, что в соответствующем фрагменте только сообщения m и s рассматриваются как существенные, а все остальные могут быть проигнорированы
- Например, выражение **ignore** {q, r} указывает, что в соответствующем фрагменте только сообщения q и r рассматриваются как несущественные
- Операнды **ignore** и **consider** могут быть объединены с другими операндами в одном прямоугольнике в качестве сокращения для вложенных фреймов. Например, **assert consider** {m, s}, **neg ignore** {q, r}.

Пример комбинированного фрагмента Игнорирование



7. Цикл (loop)

- Оператор взаимодействия **loop** специфицирует комбинированный фрагмент *Цикл* (loop), который представляет собой циклическое повторение некоторой последовательности сообщений.
- Операнд цикла повторяется несколько раз
- Дополнительное сторожевое условие может включать нижний и верхний пределы числа повторений цикла, а также некоторое логическое выражение.
- Оператор цикла имеет следующий синтаксис (БНФ):

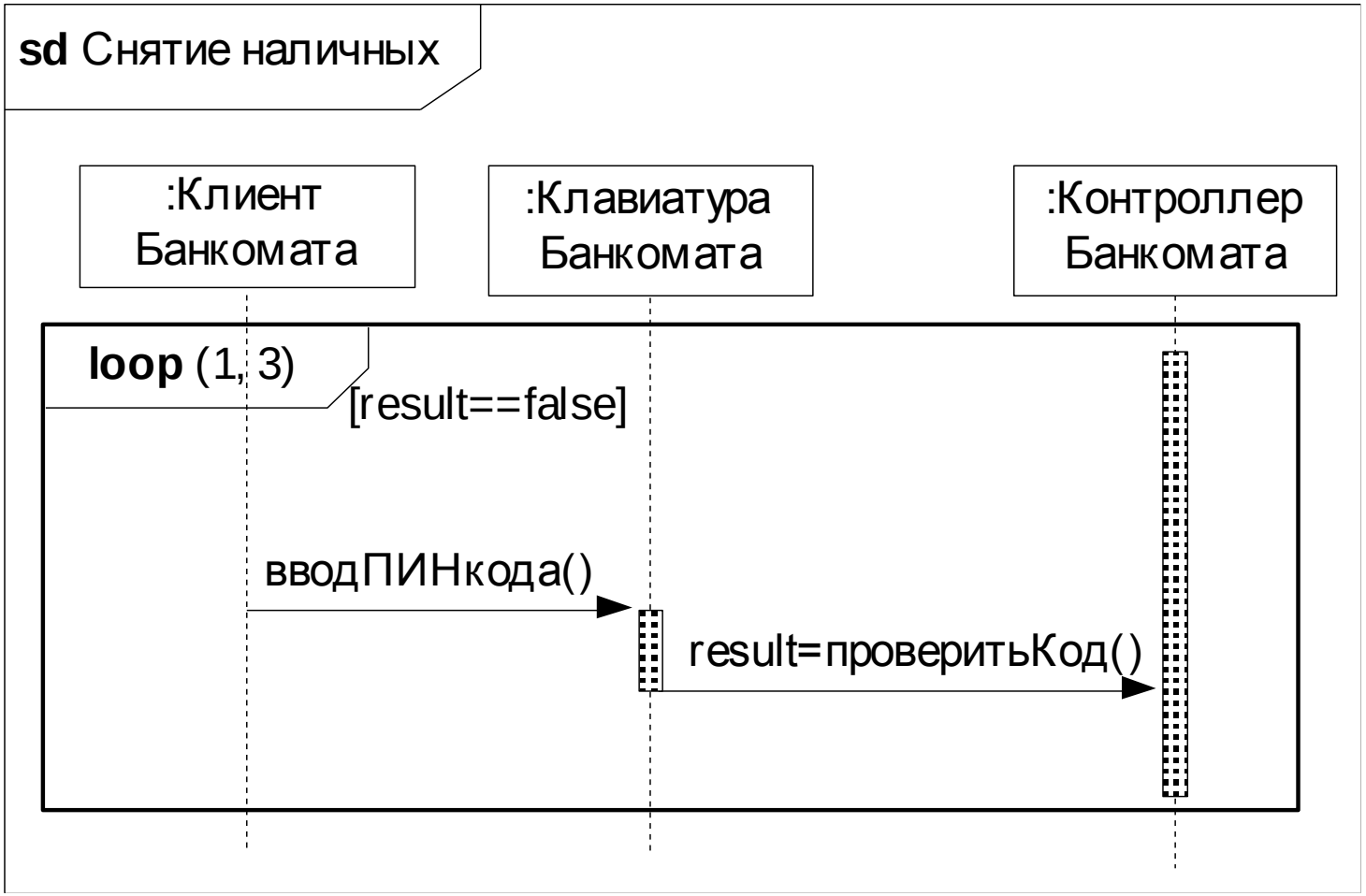
<*цикл*>::=*'loop'*['(' <*minint*> [',' <*maxint*>] ')'],

- где <*minint*>::= неотрицательное натуральное число, которое обозначает минимальное количество итераций цикла;
- <*maxint*>::= натуральное число, которое обозначает максимальное количество итераций цикла.
- Значение <*maxint*> должно быть больше или равно <*minint*> | '*'. Здесь символ '*' означает бесконечность.

Семантика цикла

- Операнд цикла всегда повторяется минимальное число раз, которое равно значению <*minint*>
- После того, как минимальное число повторений будет выполнено, проверяется логическое выражение сторожевого условия
- Если это логическое выражение принимает значение “ложь”, то выполнение цикла на этом заканчивается
- Если же логическое выражение принимает значение “истина”, а количество выполненных итераций не превышает значения <*maxint*>, то происходит еще одно выполнение цикла
- После этого снова проверяется логическое выражение сторожевого условия, аналогично процедуре выполнения минимального числа повторений.

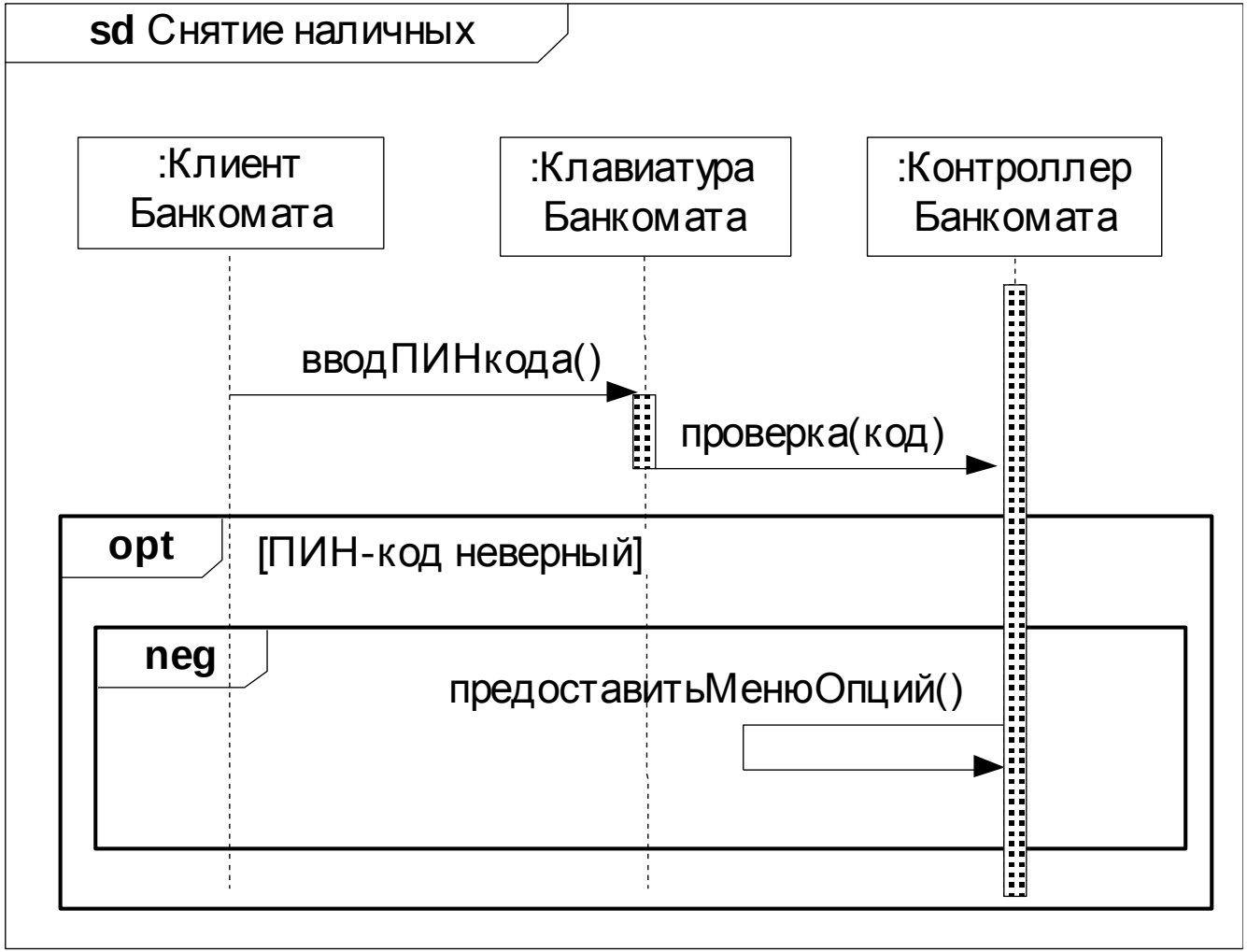
Пример комбинированного фрагмента Цикл



8. Отрицание (neg)

- Оператор взаимодействия **neg** специфицирует комбинированный фрагмент *Отрицание* (negative), который представляет траектории, которые определяются как недействительные или недопустимые
- Множество траекторий, которые определяют комбинированный фрагмент с оператором взаимодействия **neg**, равно множеству траекторий, заданных его единственным операндом
- При этом в это множество входят только недействительные или запрещенные траектории
- Все фрагменты взаимодействия, кроме Отрицания, рассматриваются в положительном смысле, т.е. они описывают траектории, которые являются допустимыми и возможными.

Пример комбинированного фрагмента Отрицание



9. Необязательный (opt)

- Оператор взаимодействия **opt** специфицирует комбинированный фрагмент *Необязательный* (option), который представляет выбор поведения, когда или выполняется единственный операнд, или вовсе ничего не выполняется
- Оператор выбора семантически эквивалентен альтернативному комбинированному фрагменту, в котором имеется один операнд с непустым содержанием, а второй операнд отсутствует.
- Необязательный комбинированный фрагмент состоит из одного операнда со сторожевым условием
- Операнд выполняется, если выполнено сторожевое условие. В противном случае операнд не выполняется.

10. Параллельный (par)

- Оператор взаимодействия **par** специфицирует комбинированный фрагмент *Параллельный* (parallel), который представляет некоторое параллельное выполнение взаимодействий своих операндов
- Наступление событий у различных операндов могут чередоваться во времени произвольным образом
- Внутри каждого операнда соблюдается порядок следования сообщений
- Каждый операнд изображается в отдельном регионе, который отделяется от других регионов пунктирной линией

11. Слабое следование (seq)

- Оператор взаимодействия **seq** специфицирует комбинированный фрагмент *Слабое следование* (weak sequencing), который состоит из нескольких операндов и представляет слабое следование поведений отдельных операндов
- Слабое следование определяется как множество траекторий со следующими свойствами:
- Порядок наступления событий в пределах каждого из операндов определяется порядком передачи сообщений во времени (сверху вниз).
- Наступление событий на различных линиях жизни у различных операндов могут происходить в произвольном порядке.
- Наступление событий на одной линии жизни у различных операндов упорядочиваются сверху вниз, т.е. наступление событий у первого операнда происходит до наступления событий у второго операнда и т.д.

12. Строгое следование (strict)

- Оператор взаимодействия **strict** специфицирует комбинированный фрагмент *Строгое следование* (strict sequencing), который состоит из нескольких операндов и представляет строгий порядок следования поведений отдельных операндов
- Данный оператор указывает, что операнды верхнего уровня комбинированного фрагмента выполняются в строго определенном порядке (сверху вниз) и не перекрываются.
- Строгий порядок следования означает, что вертикальная координата вложенных фрагментов имеет значение на всем протяжении границ комбинированного фрагмента, а не только для одной линии жизни
- Вертикальная позиция спецификации наступления события задается вертикальной позицией соответствующей точки
- Вертикальная позиция других фрагментов взаимодействия задается самой верхней вертикальной позицией соответствующих фреймов.

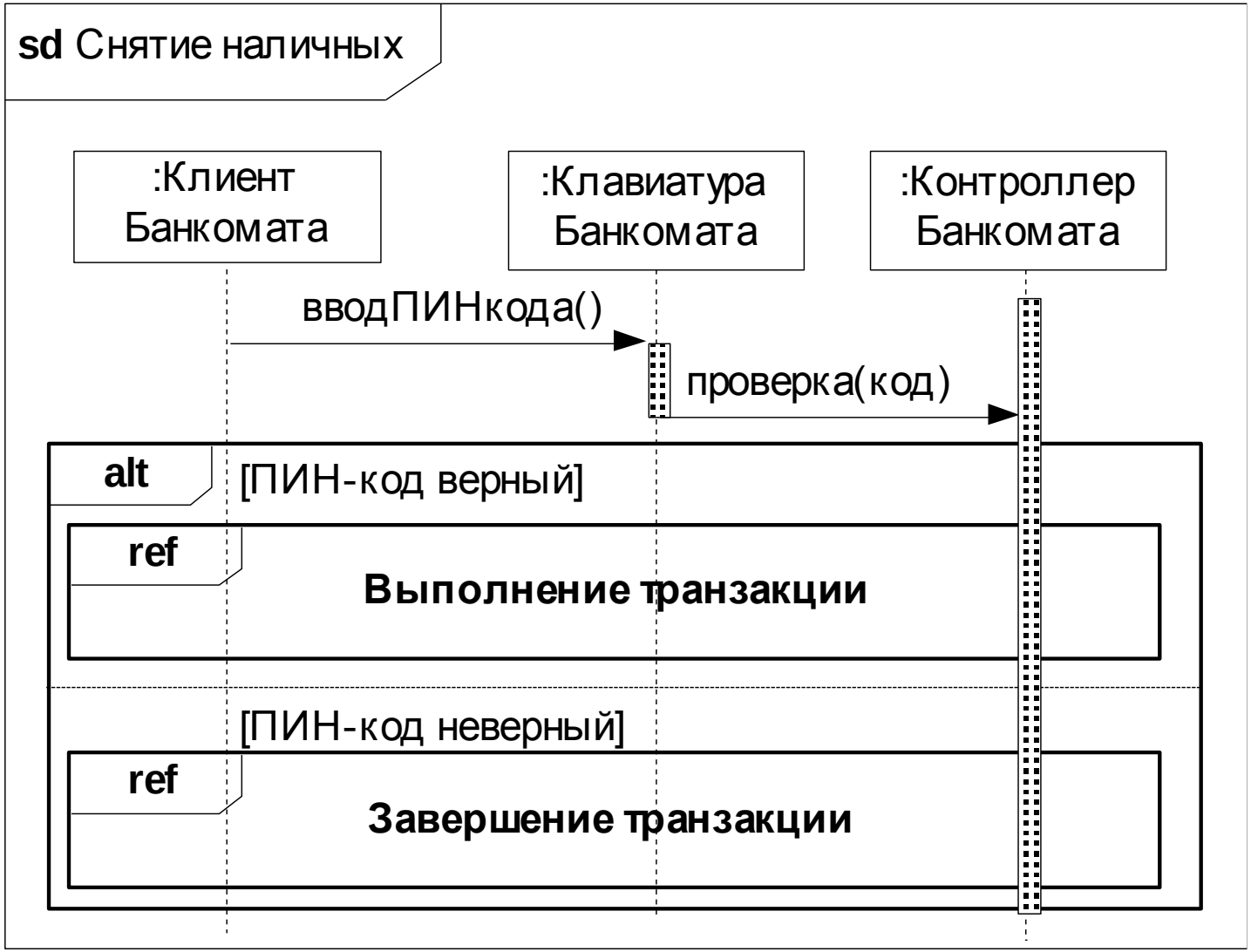
Использование взаимодействия *(interaction use)*

- — элемент модели, представляющий параметризованную ссылку на некоторое взаимодействие в контексте другого взаимодействия
- Использование взаимодействия изображается в форме фрейма комбинированного фрагмента с оператором **ref**, за которым следует полное имя использования взаимодействия
- Синтаксис полного имени использования взаимодействия (БНФ):

<имя>::=[<имя-атрибута>'='] [<использование-кооперации>'.'] <имя-взаимодействия> ['(' <io-аргумент> [',' <io-аргумент>] ')'] [':' <возвращаемое-значение>,*

- где *<io-аргумент> ::= <in-аргумент> |'out' <out-аргумент>*;
- *<имя-атрибута>* – атрибут некоторой линии жизни взаимодействия; *<использование-кооперации>* является спецификацией использования некоторой кооперации с линиями жизни данного взаимодействия

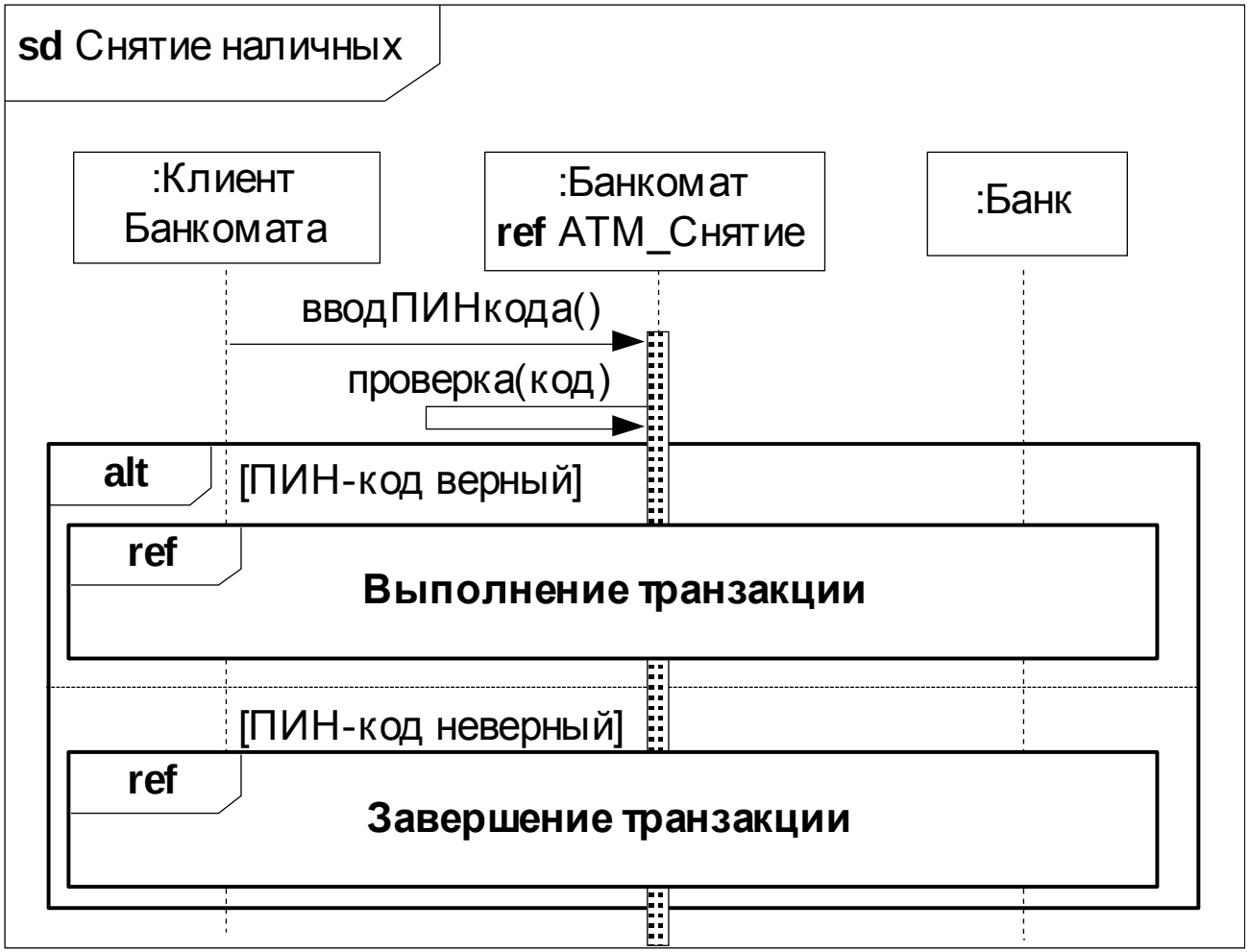
Пример использования взаимодействия



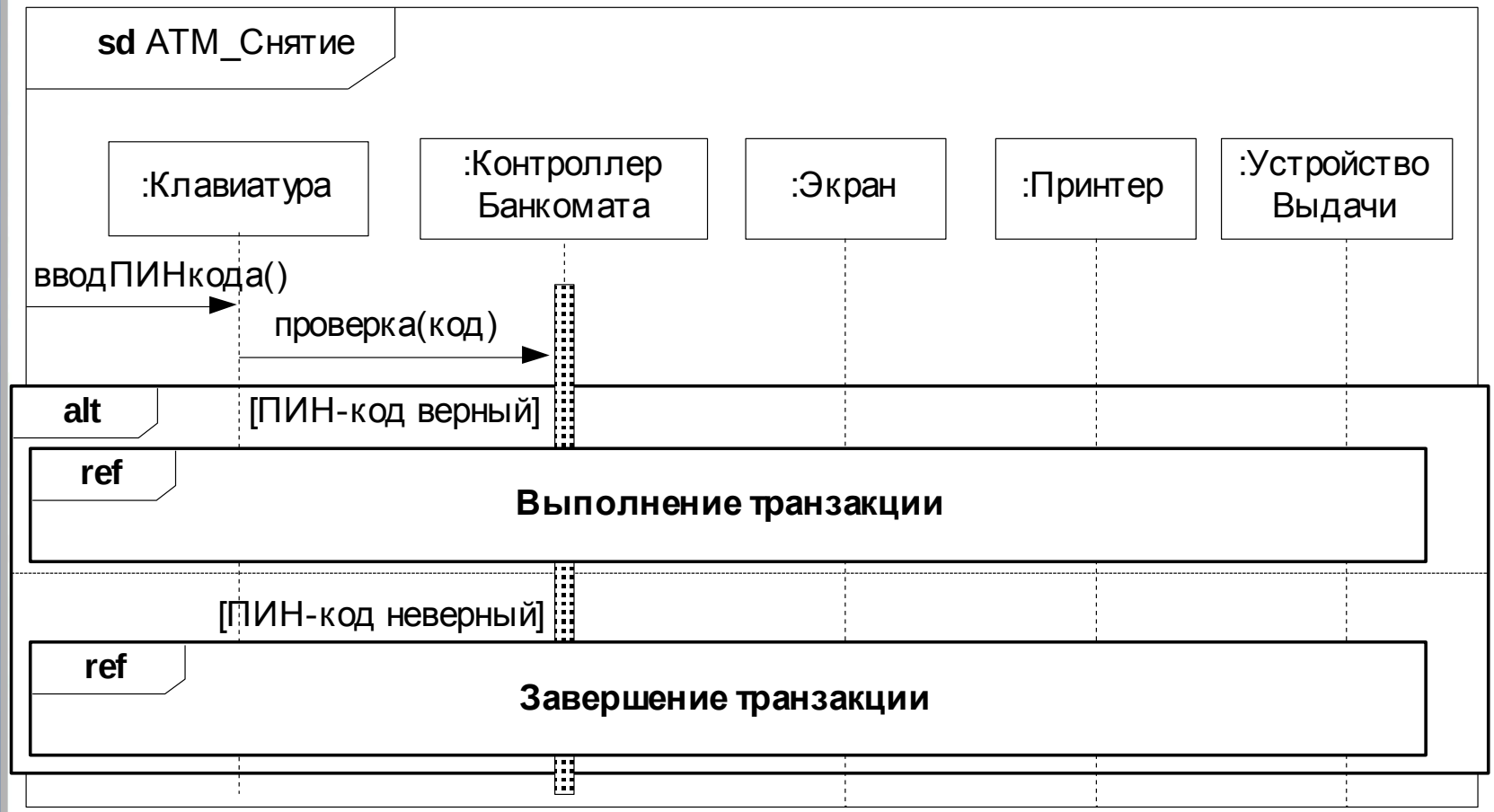
Декомпозиция части *(part decomposition)*

- - является элементом модели, предназначенным для представления внутренних взаимодействий одной из линий жизни, класс которой имеет собственную композитную структуру
- Декомпозиция части обозначается посредством ссылки в заголовке линии жизни на некоторое использование взаимодействия с помощью оператора **ref**
- Границы символов фреймов глобальных комбинированных фрагментов изображаются выходящими за пределы границ декомпозиции части
- Сообщения, которые входят или выходят из декомпозированной линии жизни интерпретируются как действительные шлюзы, которые должны соответствовать формальным шлюзам этой декомпозиции.

Пример декомпозиция части в форме ссылки в заголовке линии жизни



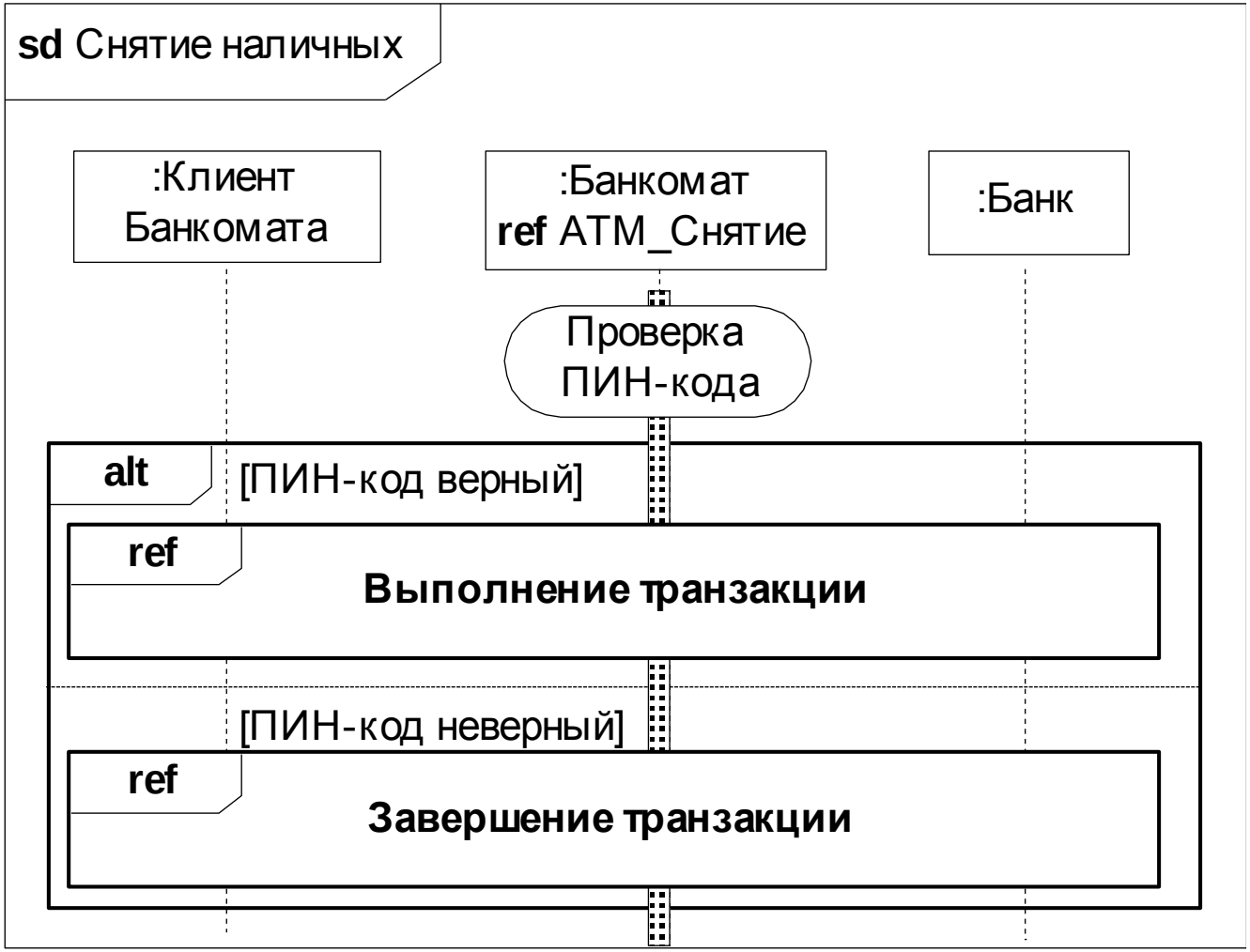
Пример диаграммы последовательности для декомпозиции части



Инвариант состояния *(state invariant)*

- - является некоторым ограничением времени выполнения, которое должно быть выполнено для отдельных участников взаимодействия
- Инвариант состояния изображается в форме символа состояния на линии жизни соответствующего участника взаимодействия
- Символ состояния представляет эквивалент ограничения, которое проверяет состояние объекта, представленного данной линией жизни
- Это может быть внутреннее состояние поведения объекта соответствующего класса или некоторое внешнее состояние, основанное на представлении “черный ящик” для данной линии жизни.

Пример представления инварианта состояния в форме символа состояния



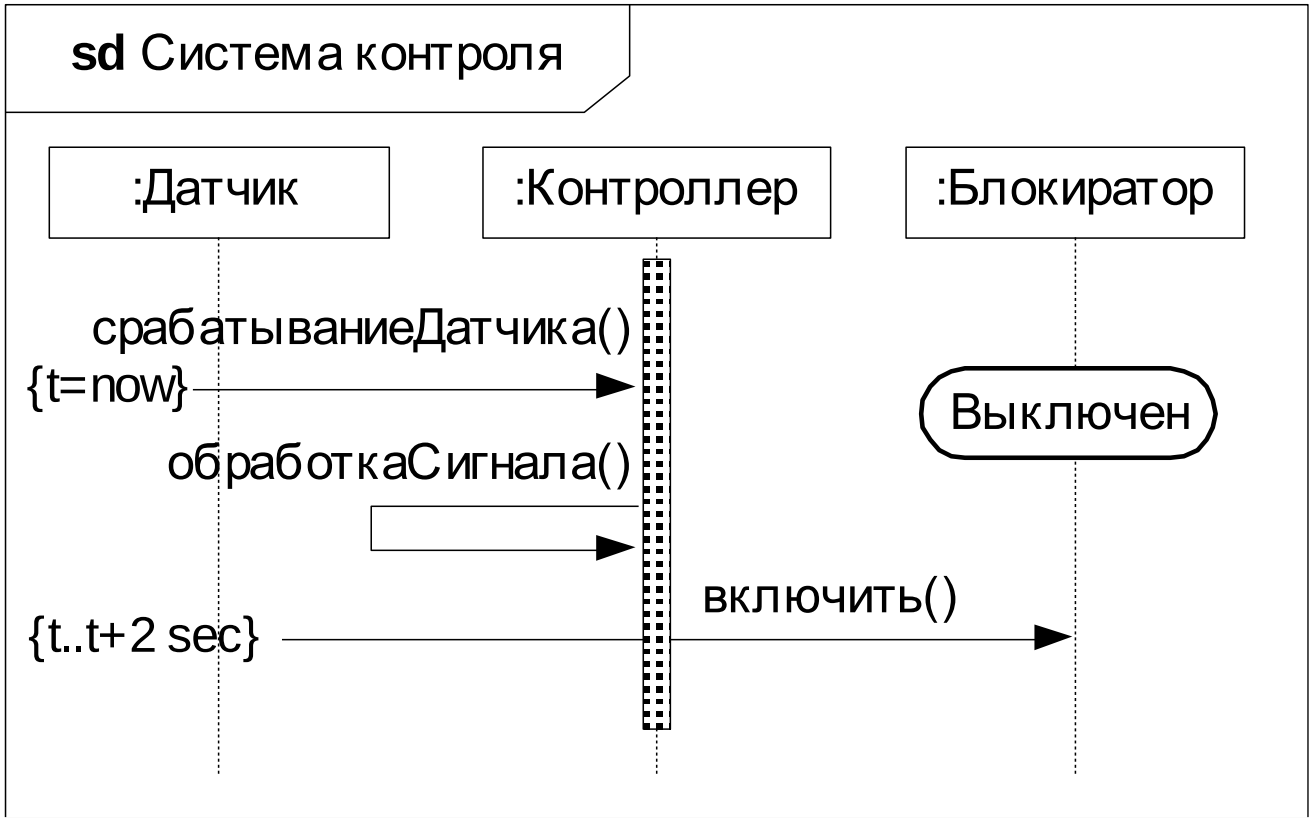
Пример представления инварианта состояния в форме ограничения

Продолжение *(continuation)*

- - представляет собой некоторую метку, которая позволяет разбивать операнды комбинированного фрагмента Альтернативы на две и более части и комбинировать их траектории в различных фреймах
- Метки продолжения интуитивно представляют промежуточные точки в потоке управления комбинированного фрагмента Альтернативы и могут находиться в начале или конце этого фрагмента
- Продолжение имеет смысл только в контексте комбинированного фрагмента Альтернативы и слабого следования
- Продолжение изображается символом состояния, но этот символ, в отличие от инварианта состояния, может покрывать более чем одну линию жизни

Временное ограничение *(time constraint)*

- - представляет собой специальное ограничение, записанное в форме временного интервала.



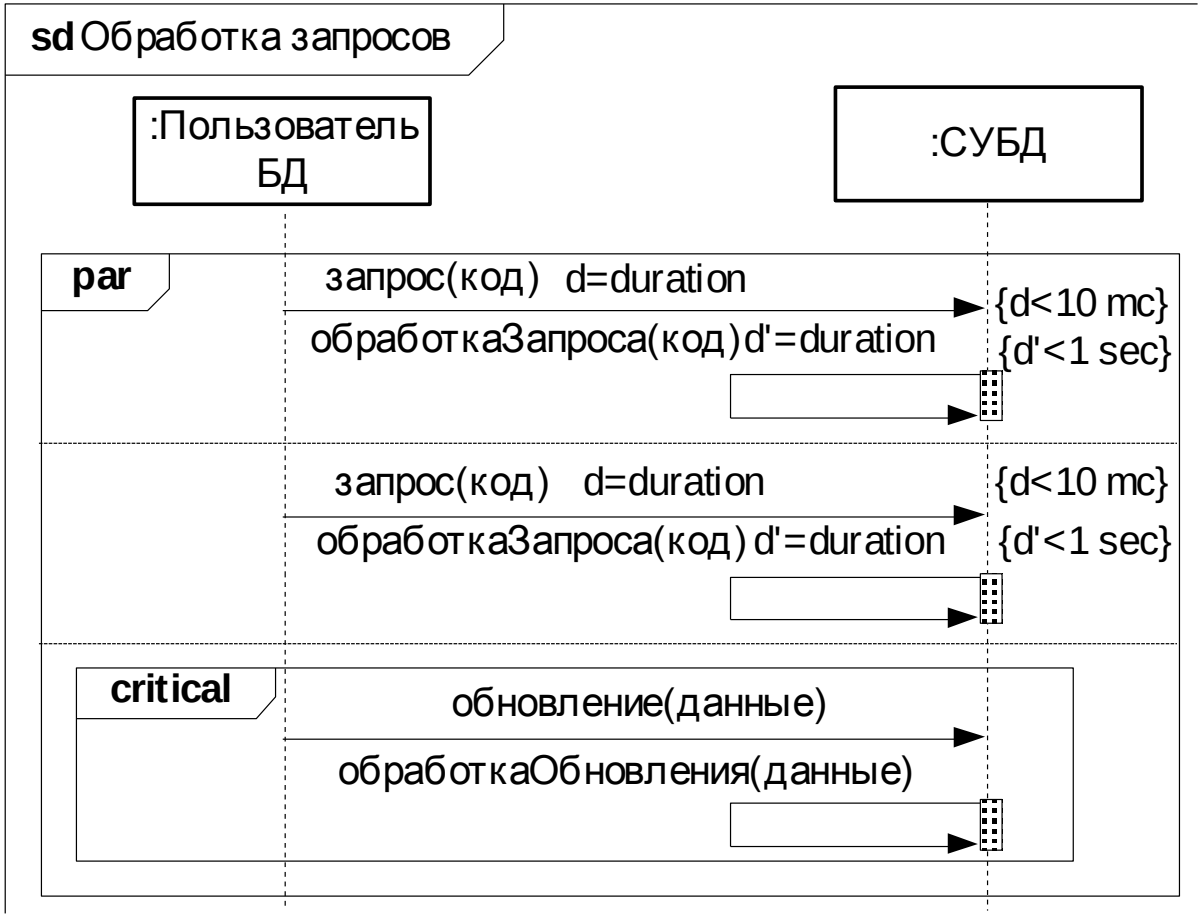
Продолжительность *(duration)*

- - специфицирует временное расстояние между двумя временными выражениями, которые соответствуют двум моментам времени
- *Интервал продолжительности (duration interval)* определяет диапазон между двумя продолжительностями
- *Действие наблюдения продолжительности (duration observation action)* определяется как действие, которое наблюдает продолжительность во времени и записывает это значение в некоторую структурную характеристику
- Формальный синтаксис действие наблюдения продолжительности (БНФ):

  <действие-наблюдение-продолжительности>::= <имя-атрибута>‘=duration’

Ограничение на продолжительность *(duration constraint)*

- - определяет ограничение, которое ссылается на некоторый интервал продолжительности



Самостоятельное задание №5

- Выполнить текущее тестирование: вопросы 21-23
- Разработать диаграмму последовательности для представления полной функциональности ATM
  - Изобразить линии жизни соответствующих классов
  - Изобразить необходимые комбинированные фрагменты
  - Изобразить необходимые сообщения между линиями жизни