Диаграмма последовательности

UML-диаграммы

Диаграмма последовательности

Диаграмма последовательности (*Sequence Diagram)* служит для представления взаимодействия между элементами модели (классификаторами, объектами, другими сущностями) в форме последовательности сообщений и соответствующих им событий на линиях жизни сущностей.

Масштаб осей времени на диаграмме последовательности не указывается, поскольку эта диаграмма предназначена для представления временной упорядоченности сообщений в терминах «раньше-позже».

Взаимодействие

Под взаимодействием сущностей (*interaction*) понимается такое поведение, при котором происходит обмен информацией между сущностями. Обычно, взаимодействуют друг с другом объекты. Их дальше и будем иметь ввиду.

Взаимодействие объектов на диаграмме последовательности изображается в виде фреймов (*frame*). В данном случае под фреймом можно понимать просто прямоугольник с непрерывными сторонами. В верхнем левом углу фрейма изображается небольшой пятиугольник, в котором помещается ключевое слово *sd*, за которым следует имя взаимодействия и, возможно, его параметры.

Линии жизни

Линия жизни объекта *(lifeline)* представляет собой жизненный цикл одного участника взаимодействия (взаимодействующего объекта).

Порядок наступления событий вдоль линий жизни важен, именно он задает последовательность, в которой эти события происходят. Оси времени на линиях жизни всегда направлены «сверху вниз». Абсолютные расстояния между наступлениями событий на осях времени не имеют значения.

Другими словами, оси времени на диаграмме последовательности служат только для спецификации порядка наступления событий, но не для отсчета собственно времени. Это необходимо помнить при разработке диаграмм последовательности.

Диаграмма последовательности

Идентификатор линии жизни объекта изображается внутри прямоугольника в следующем формате (БНФ):

<идентификатор-линии-жизни>::= ([<имя-взаимод.-элемента> [‘[‘<селектор>‘] ’]] [: <имя-класса>] [декомпозиция]) | ‘self’

где *<селектор>::= <выражение>*

<декомпозиция>::=‘ref’ <идентификатор-взаимодействия> [‘strict’]

Здесь *<имя-класса>* является типом, на который ссылается представленный объект.

Если *<именем-класса>* является ключевое слово *self*, то такая линия жизни представляет объект классификатора, который владеет данным взаимодействием.

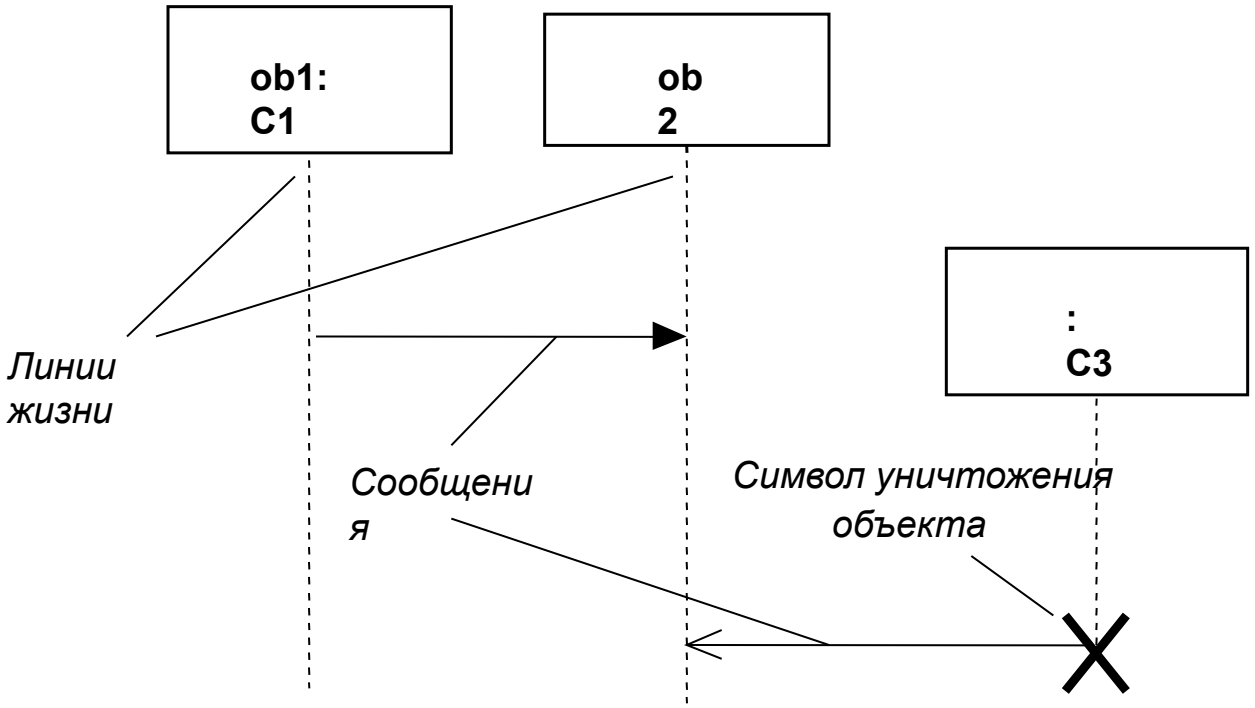
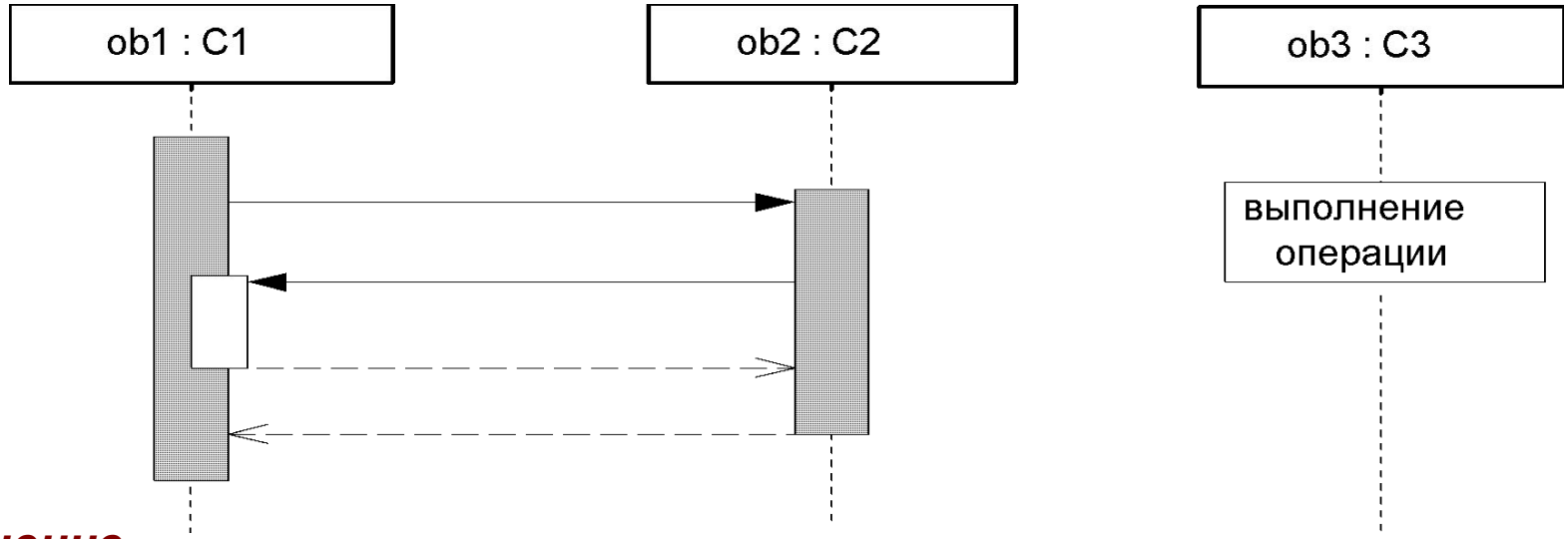


Диаграмма последовательности

Спецификация выполнения

Execution Specification предназначена для моделирования состояния активности линии жизни в описываемом взаимодействии. Иногда спецификацию выполнения называют фокусом управления (при синхронном взаимодействии).



Сообщение

Message - элемент модели, предназначенный для представления отдельной коммуникации между линиями жизни взаимодействующих объектов.

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>]|‘ -’*

Диаграмма последовательности

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:

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.

ответ (*reply*) от вызова метода, изображается пунктирной линией с открытой стрелкой в форме буквы “V”.

Сообщение о создании объекта (*object creation*) также изображается пунктирной линией с открытой стрелкой в форме буквы “V”:

Сообщения, сигналы и ответы образуют **траектории взаимодействия**.

Диаграмма последовательности

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. Такие сообщения не должны представляться на диаграмме последовательности.

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.

Диаграмма последовательности

Сигнал

Signal – представляет собой спецификацию асинхронной коммуникации между линиями жизни.

Событие сигнала (*signal event*) представляет собой факт приема на линии жизни объекта 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 взаимодействия объектов.

Операнд взаимодействия (*interaction operand*) есть отдельный неделимый квант взаимодействия. Комбинированный фрагмент состоит из операндов взаимодействия.

Ограничение взаимодействия (*interaction constraint*) есть логическое выражение, которое выступает в роли сторожевого условия некоторого операнда взаимодействия в комбинированном фрагменте.

Диаграмма последовательности

Графическое изображение комбинированного фрагмента.

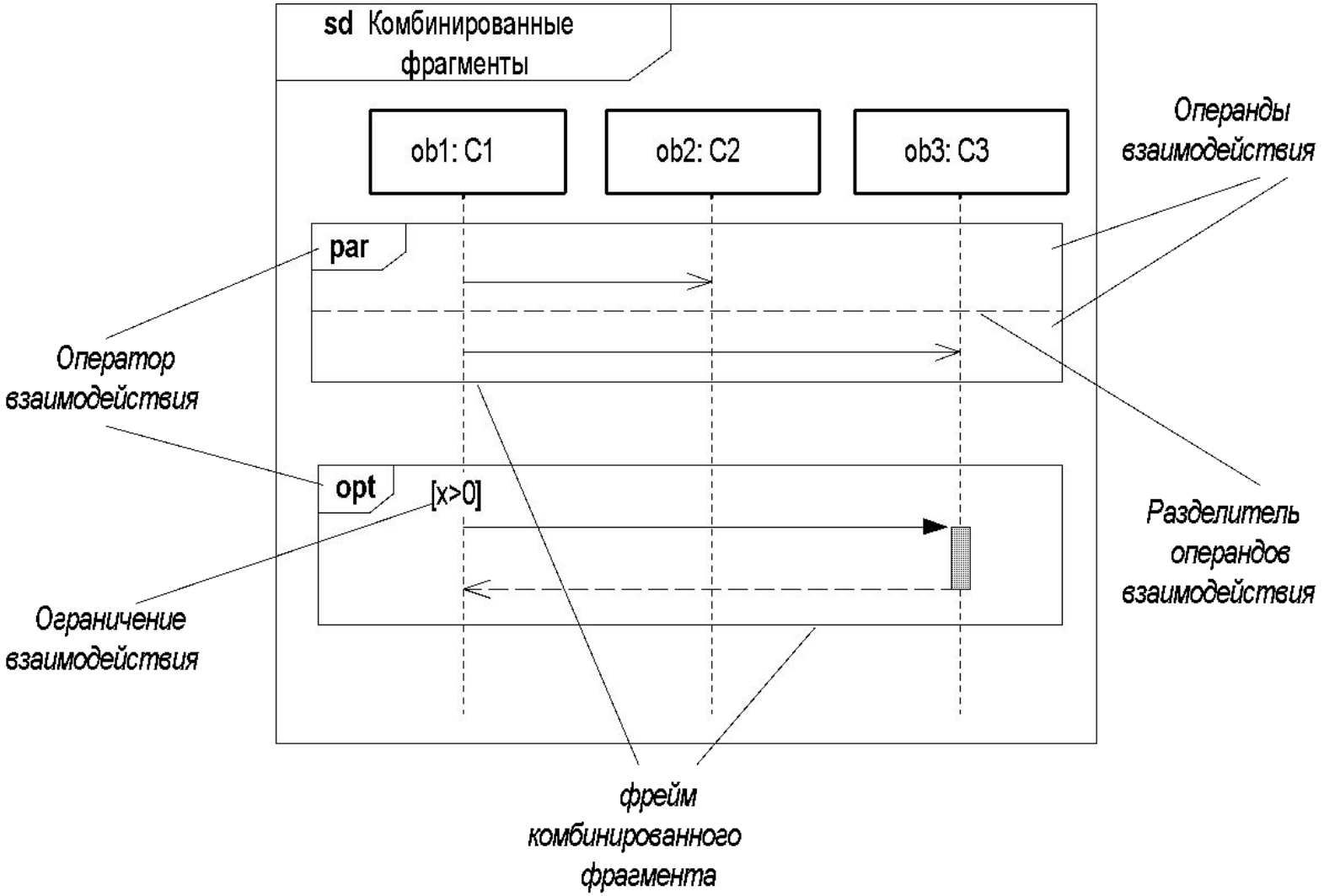


Диаграмма последовательности

Операторы 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***

Альтернатива (alt)

Оператор взаимодействия ***alt*** специфицирует комбинированный фрагмент «*Альтернатива*» (*alternative*), который представляет некоторый выбор поведения.

Может быть выбран не более, чем один операнд.

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 «истина».

Операнд, помеченный сторожевым условием [*else*], обозначает отрицание дизъюнкции всех других сторожевых условий этого комбинированного фрагмента.

Диаграмма последовательности

Пример комбинированного фрагмента «*Альтернатива*».

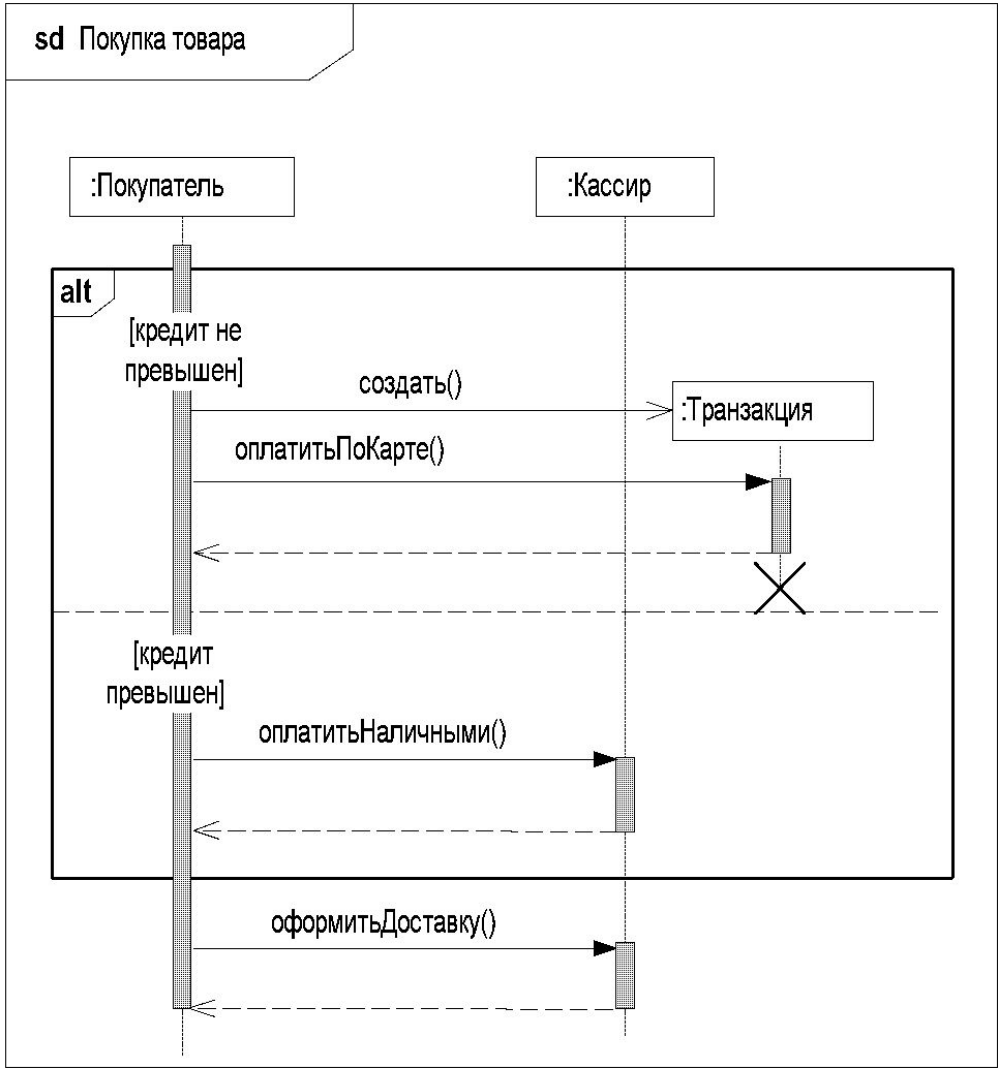


Диаграмма последовательности

Утверждение (assert)

Оператор взаимодействия ***assert*** специфицирует комбинированный фрагмент «*Утверждение*» (*assertion*). Специфицирует единственно возможное поведение в данный момент взаимодействия объектов.

Любое другое поведение при взаимодействии объектов было бы **ошибочным**.

Оператор ***assert*** применяется для обозначения того, что некоторое поведение **должно иметь место** в данной точке взаимодействия.

Комбинированный фрагмент «*Утверждение*» часто объединяется с операторами ***ignore*** или ***consider***.

Завершение (break)

Оператор взаимодействия ***break*** специфицирует комбинированный фрагмент «*Завершение*» (*break*), который представляет некоторый сценарий завершения взаимодействия объектов.

Этот сценарий выполняется вместо оставшейся части фрагмента взаимодействия, который содержит этот операнд.

Обычно комбинированный фрагмент «*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 комбинированного фрагмента «*Завершение*» принимает значение «*ложь*», то комбинированный фрагмент «*Завершение*» игнорируется, и выполняется оставшаяся часть фрагмента взаимодействия.

Диаграмма последовательности

Пример комбинированного фрагмента «*Завершение*».

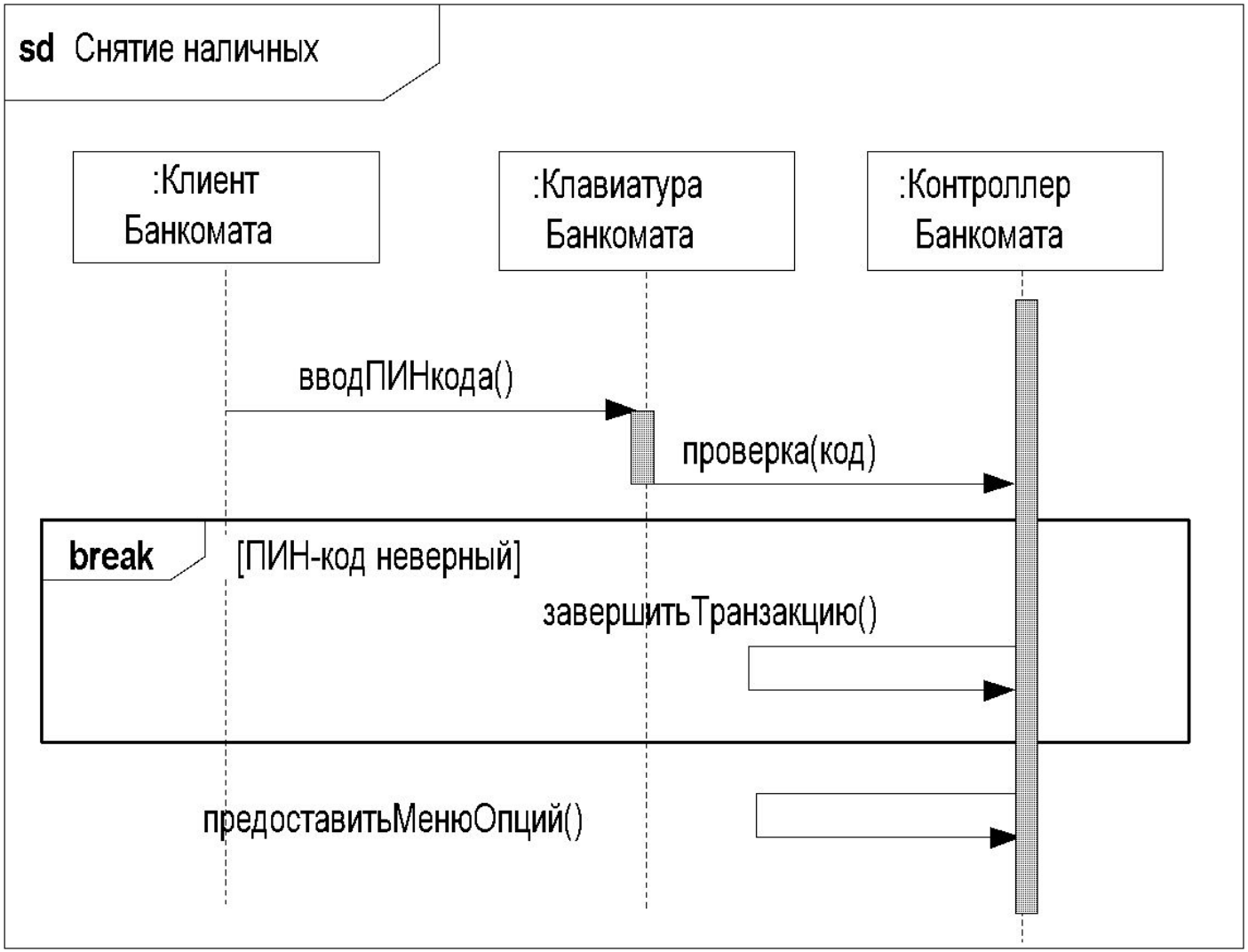


Диаграмма последовательности

Критический регион (critical)

Оператор взаимодействия ***critical*** специфицирует комбинированный фрагмент «*Критический регион*» (*critical region*), траектории которого не могут прерываться другими событиями на тех линиях жизни, которые этот регион покрывает.

«*Критический регион*» рассматривается как неделимый при определении множества возможных траекторий диаграммы, которая 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 ***par***.

Рассмотрение(consider)

Оператор взаимодействия ***consider*** специфицирует комбинированный фрагмент «*Рассмотрение*» (*consider*), в котором изображены те типы сообщений, которые рассматриваются как существенные (т.е. будут обрабатываться) в данном фрагменте.

Любые другие сообщения, которые не изображены в этом фрагменте, будут проигнорированы. Это может понадобится, например, при тестировании ПО.

Для фрагмента «*Рассмотрение*» используется нотация фрейма с оператором, в качестве которого используется ключевое слово *consider*.

Список обрабатываемых сообщений следует за операндом и заключается в фигурные скобки согласно следующему формату:

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

Диаграмма последовательности

Пример комбинированного фрагмента «*Критический регион*».

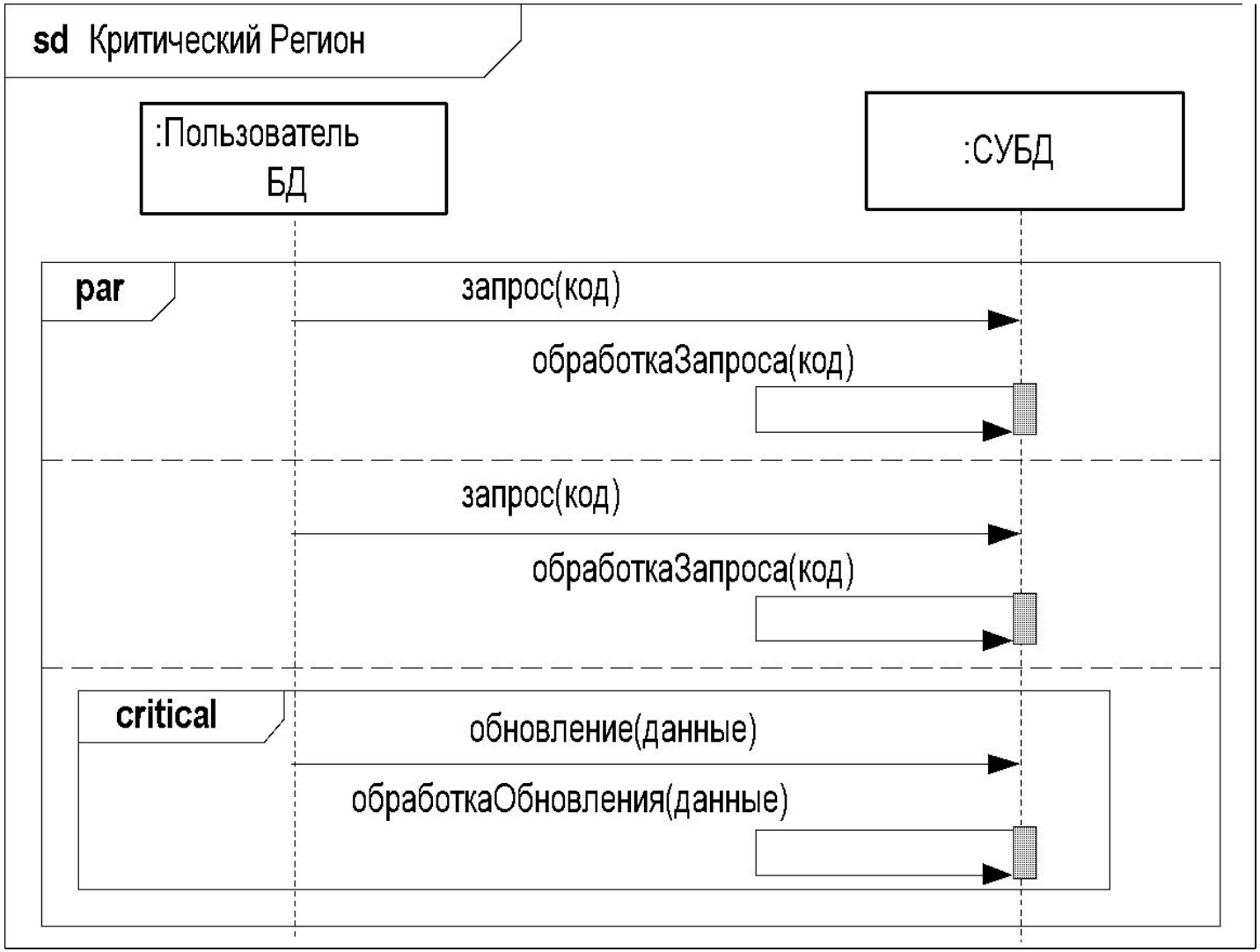


Диаграмма последовательности

Игнорирование (ignore)

Оператор взаимодействия ***ignore*** специфицирует комбинированный фрагмент «*Игнорирование*» (*ignore*), в котором в котором представлены те сообщения, которые должны быть проигнорированы в данном фрагменте (***consider*** наоборот).

Представленные типы сообщений рассматриваются как несущественные, хотя они могут появляться в траекториях при выполнении соответствующего фрагмента.

Для фрагмента «*Игнорирование*» используется нотация фрейма с оператором, в качестве которого используется ключевое слово *ignore.*

Список игнорируемых сообщений следует за операндом и заключается в фигурные скобки согласно следующему формату:

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

Примеры комбинированных фрагментов «Рассмотрение» и «Игнорирование»

Выражение *consider {m, s}* указывает, что в соответствующем фрагменте только сообщения *m* и *s* рассматриваются как существенные, а все остальные будут проигнорированы.

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, чтобы избежать излишнего вложения фреймов.

Например: *assert consider {m, s}, neg ignore {q, r}.*

Диаграмма последовательности

Пример комбинированного фрагмента «*Игнорирование*».

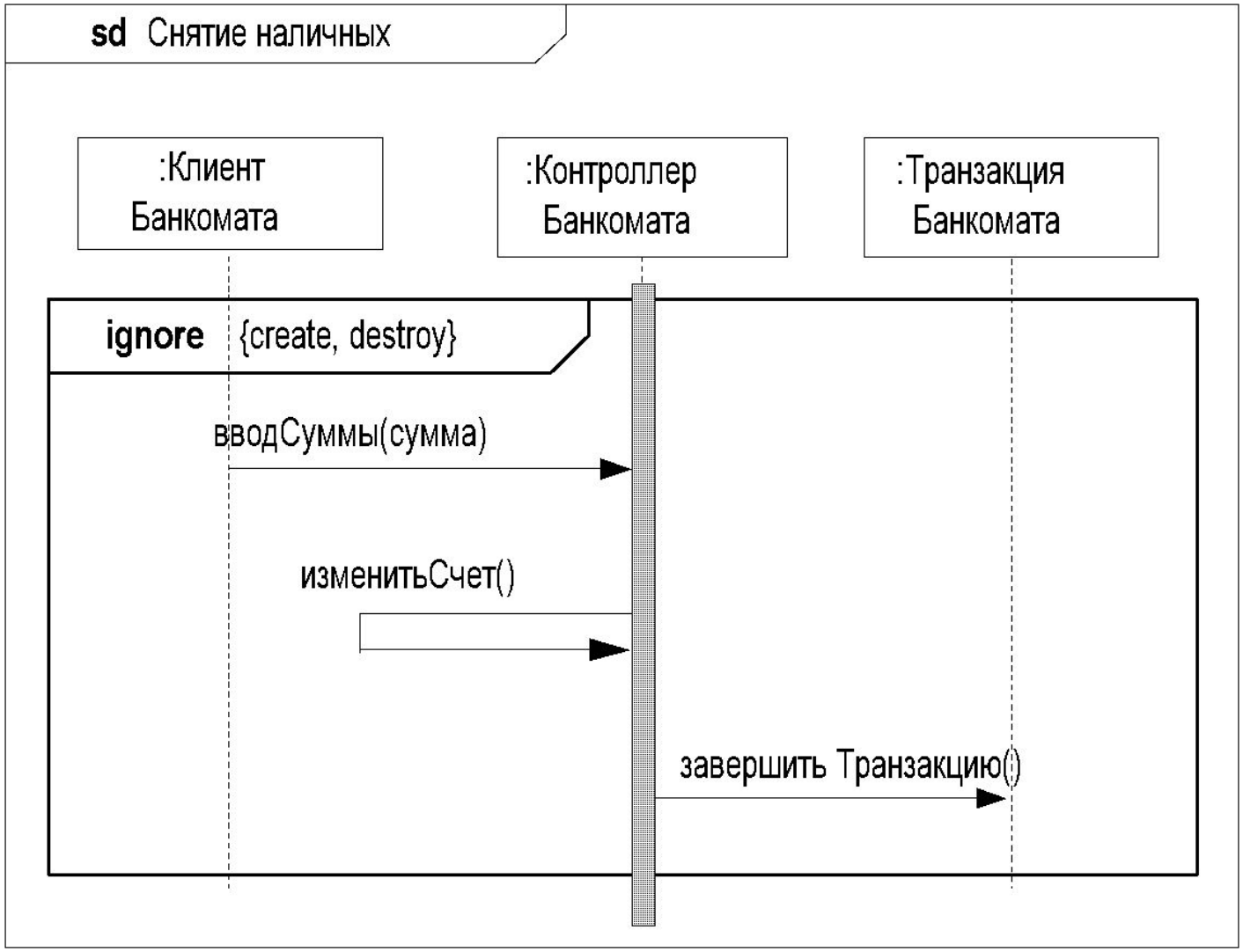


Диаграмма последовательности

Цикл (loop)

Оператор взаимодействия ***loop*** специфицирует комбинированный фрагмент «*Цикл*» (*loop*), который представляет собой циклическое повторение некоторой последовательности сообщений.

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.

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.

Диаграмма последовательности

Пример комбинированного фрагмента «*Цикл*».

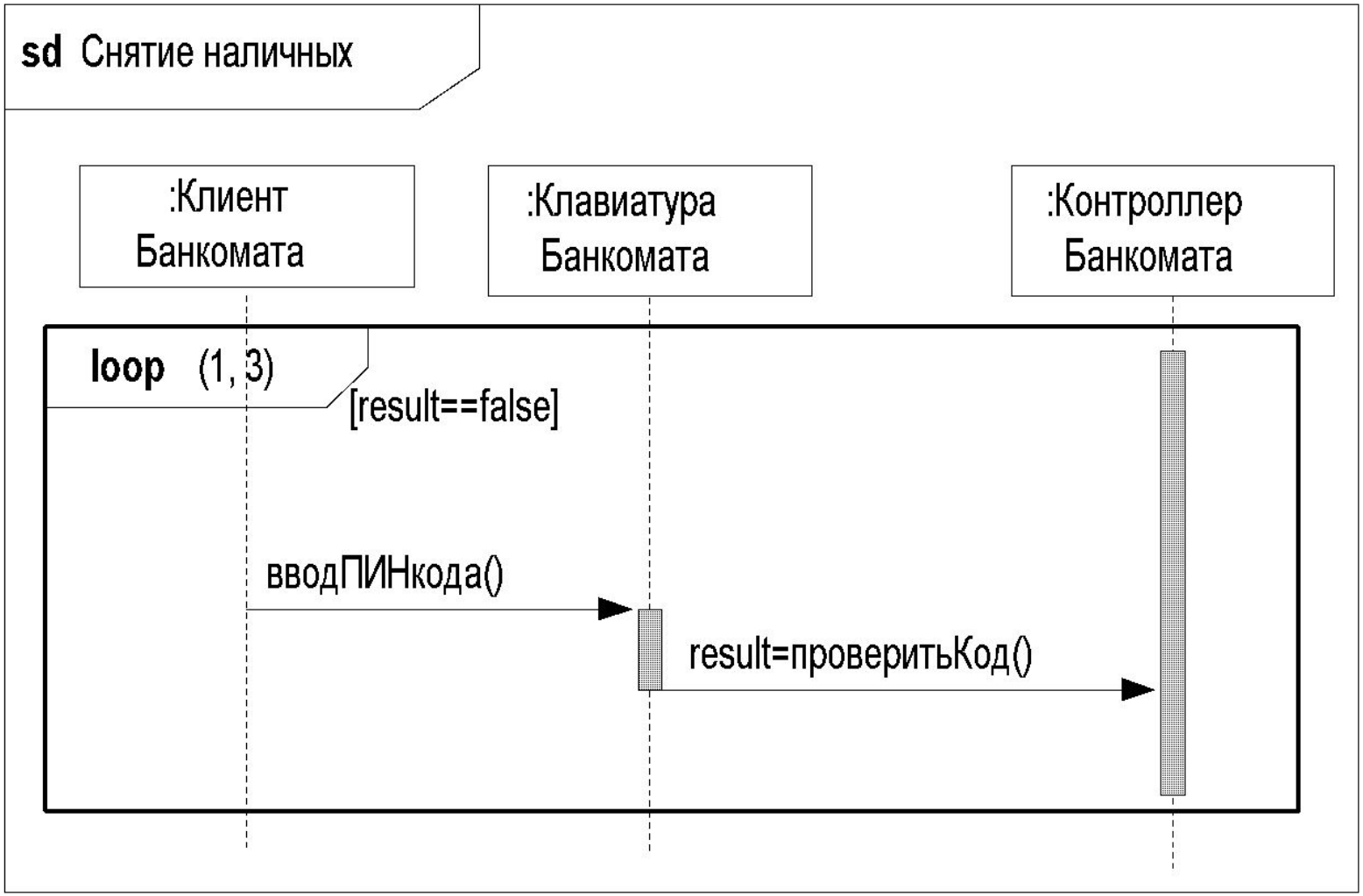


Диаграмма последовательности

Отрицание (neg)

Оператор взаимодействия ***neg*** специфицирует комбинированный фрагмент «*Отрицание*» (*negative*). Специфицирует **ошибочное** поведение в данный момент взаимодействия объектов.

Оператор ***neg*** применяется для обозначения того, что некоторое поведение **не должно иметь место** в данной точке взаимодействия.

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 взаимодействия ***opt*** семантически эквивалентен комбинированному фрагменту «*Альтернатива*», в котором имеется один операнд с непустым содержанием, а второй операнд отсутствует.

Комбинированный фрагмент «*Необязательный*» состоит из одного операнда со сторожевым условием

Операнд выполняется, если выполнено сторожевое условие. В противном случае операнд не выполняется.

Диаграмма последовательности

Пример комбинированного фрагмента «*Отрицание*».

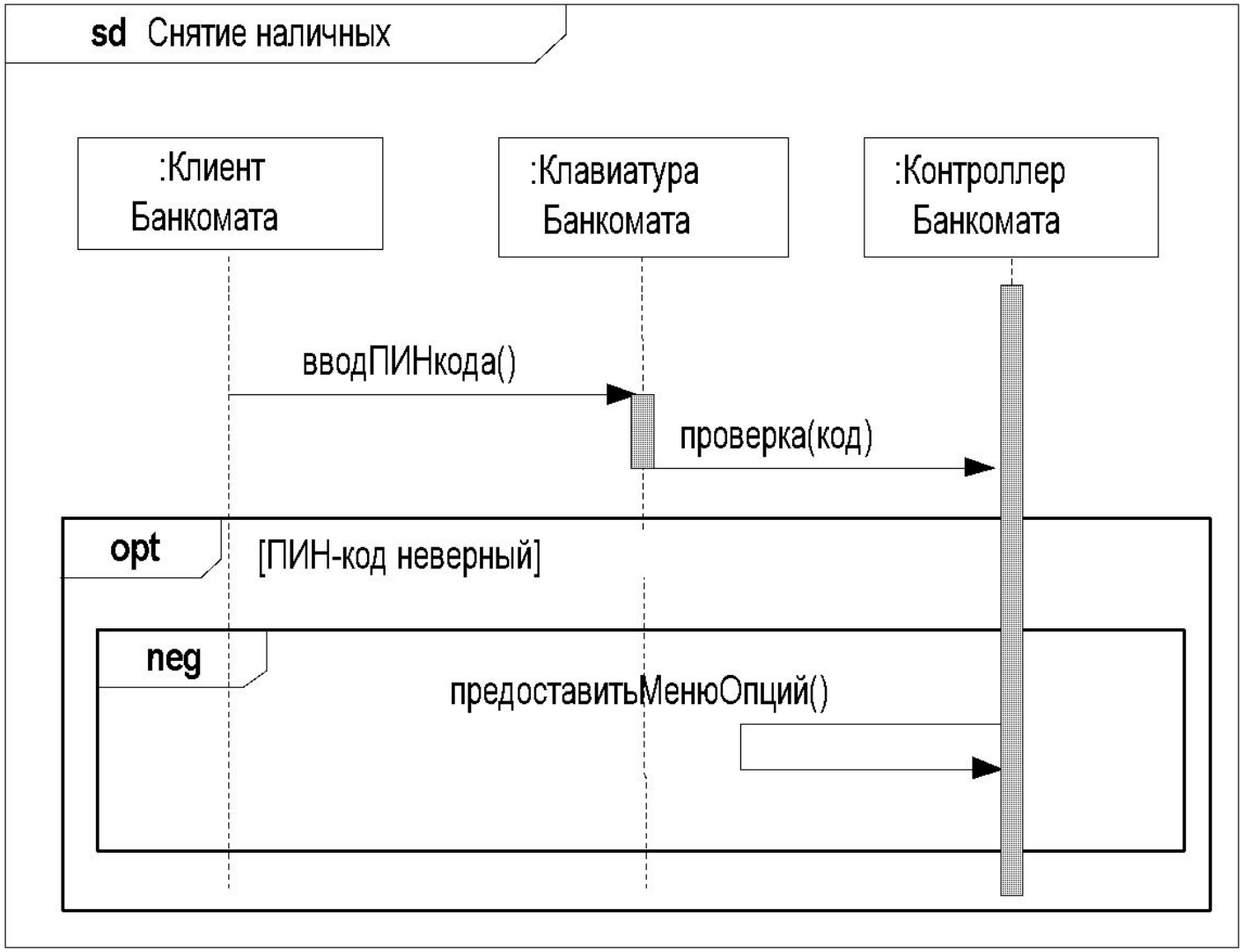


Диаграмма последовательности

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.

Слабое упорядочение (seq)

Оператор взаимодействия ***seq*** специфицирует комбинированный фрагмент «*Слабое упорядочение*» (*weak sequencing*) и состоит из нескольких операндов.

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.

Операнды выполняются параллельно, если последовательность поступления событий на одну линию жизни от разных операндов такая же, как и последовательность самих операндов.

Наступление событий на различных линиях жизни у различных операндов может происходить в произвольном порядке.

Диаграмма последовательности

Строгое упорядочение (strict)

Оператор взаимодействия ***strict*** специфицирует комбинированный фрагмент «*Строгое упорядочение*» (*strict sequencing*), который состоит из нескольких операндов. Операнды верхнего уровня не перекрываются.

Операнды выполняются в строгой последовательности (сверху вниз).

Порядок наступления события задается порядком операндов, с учетом возможной вложенности фрагментов.

Строгость упорядочения означает, что события должны быть упорядочены на всех линиях жизни, которые покрываются комбинированным фрагментом «*Строгое упорядочение*».

При слабом упорядочении (***strict)*** наступление событий на различных линиях жизни у различных операндов могжет происходить в произвольном порядке.

Важное замечание. В нашей рабочей версии *Visual Paradigm* реализованы не все комбинированные фрагменты (из 12 рассмотренных). Поэтому будьте готовы к тому, что диаграмму последовательности придется, может быть, частично «дорисовывать» вручную.

Диаграмма последовательности

Использование взаимодействия

Interaction use – элемент модели, представляющий параметризованную ссылку на некоторое взаимодействие в контексте другого взаимодействия (**вспомним шаблоны классов**).

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.

Диаграмма последовательности

Пример использования взаимодействия.

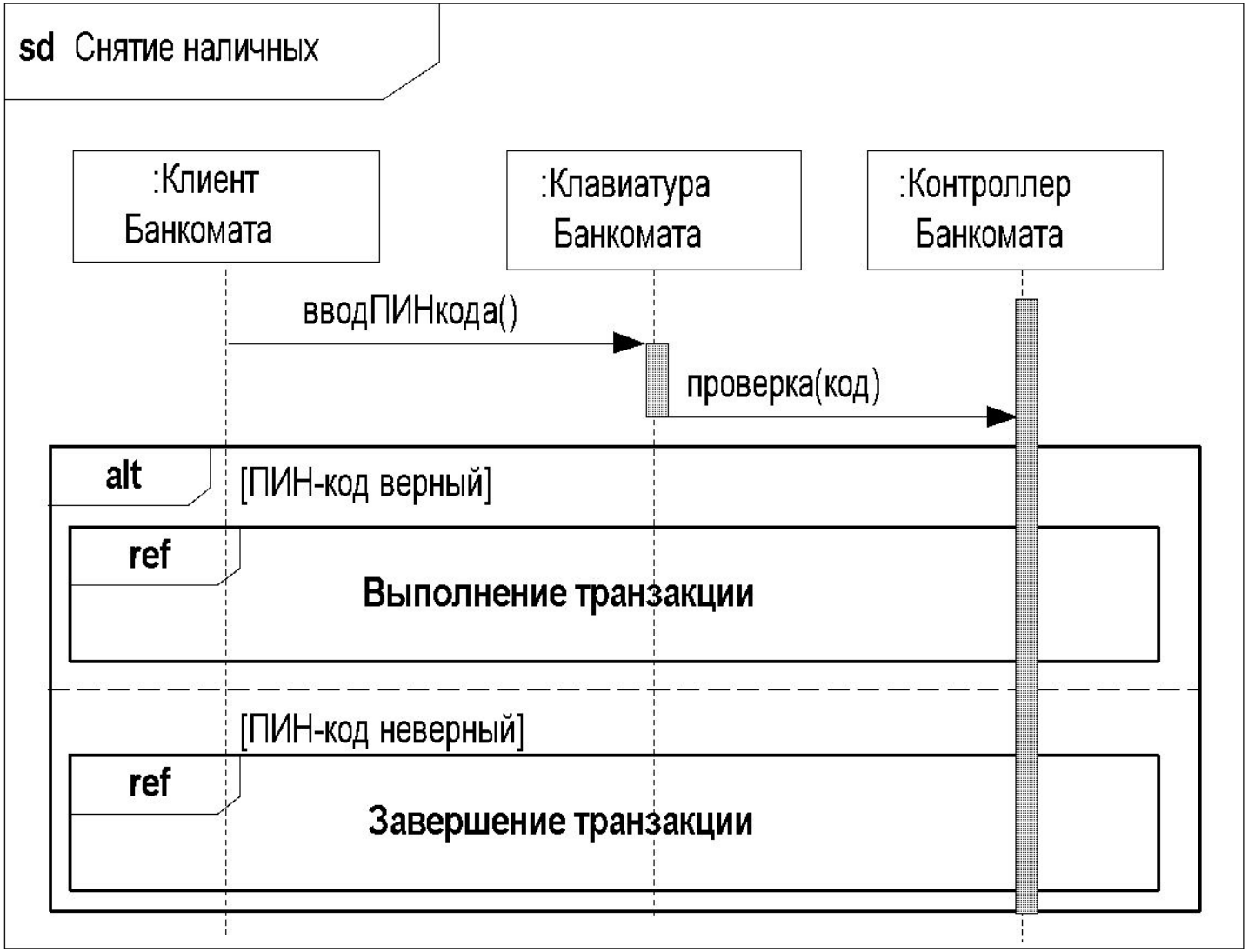


Диаграмма последовательности

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 декомпозированной линии жизни должны соответствовать сообщениям исходной (не декомпозированной) линии жизни.

Диаграмма последовательности

Пример декомпозиция части в форме ссылки в заголовке линии жизни.

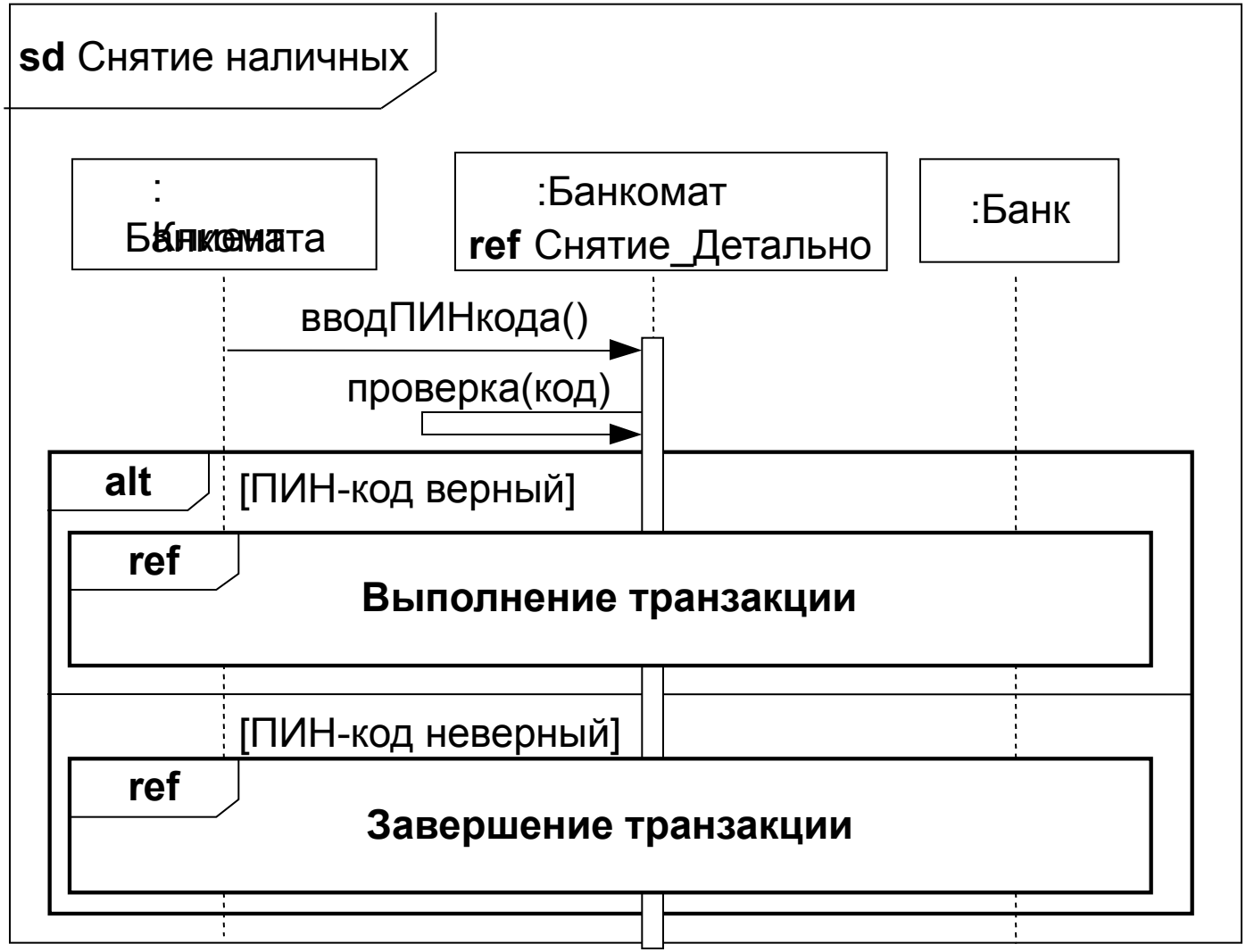


Диаграмма последовательности

Пример диаграммы последовательности для декомпозиции части.

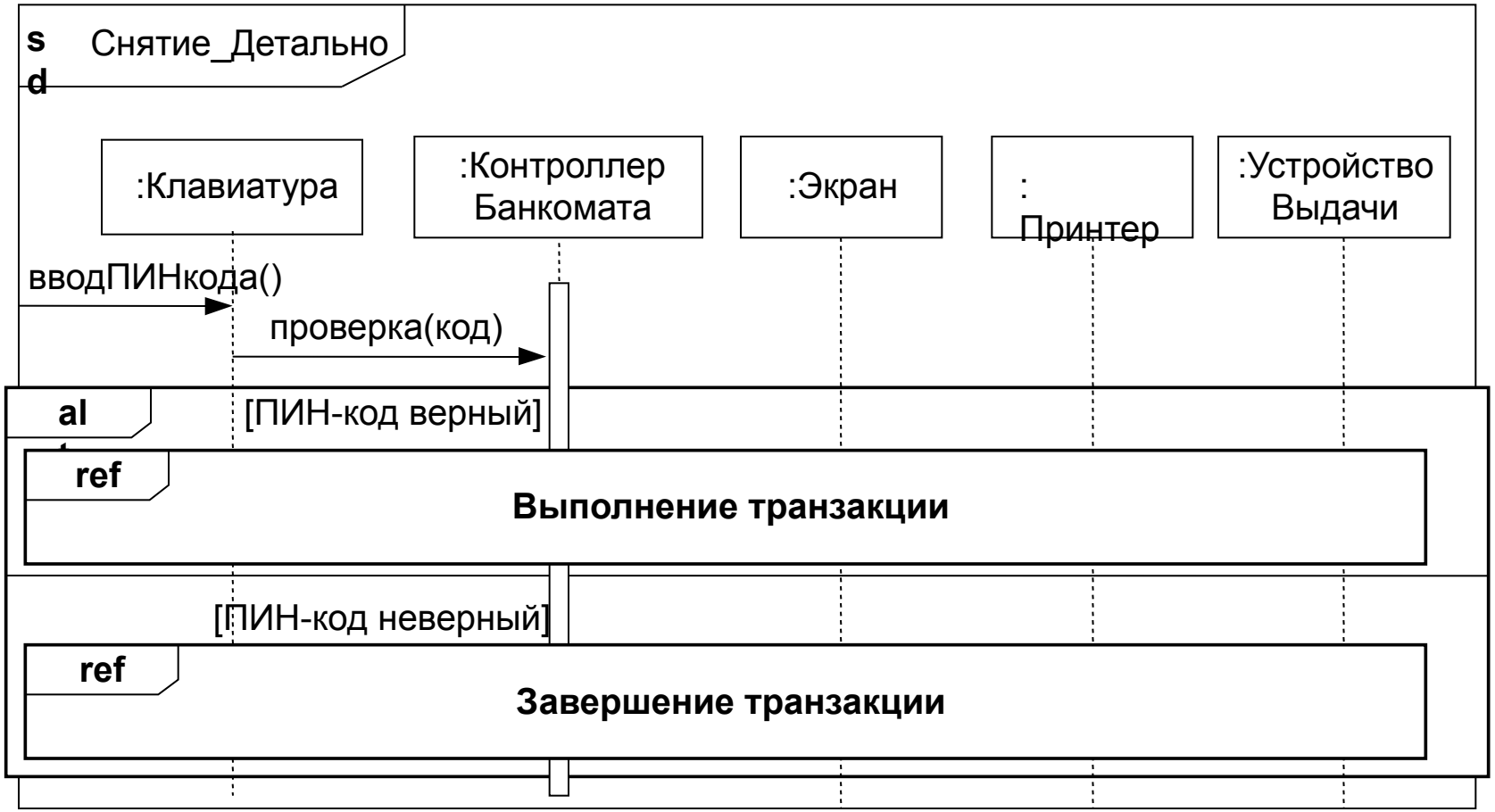


Диаграмма последовательности

Инвариант состояния

State invariant – является некоторым ограничением, 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.

Продолжение

Continuation - представляет собой некоторую метку, которая позволяет разбивать операнды комбинированного фрагмента «*Альтернатива*» на две и более части и комбинировать их траектории в различных фреймах.

Метки продолжения интуитивно представляют промежуточные точки в потоке управления комбинированного фрагмента «*Альтернатива*» и могут находиться в начале или конце этого фрагмента.

Продолжение имеет смысл только в контексте комбинированного фрагмента «*Альтернатива*» и слабого упорядочения.

Продолжение изображается символом состояния, но этот символ, в отличие от инварианта состояния, может покрывать более чем одну линию жизни.

Диаграмма последовательности

Пример представления инварианта состояния в форме символа состояния.

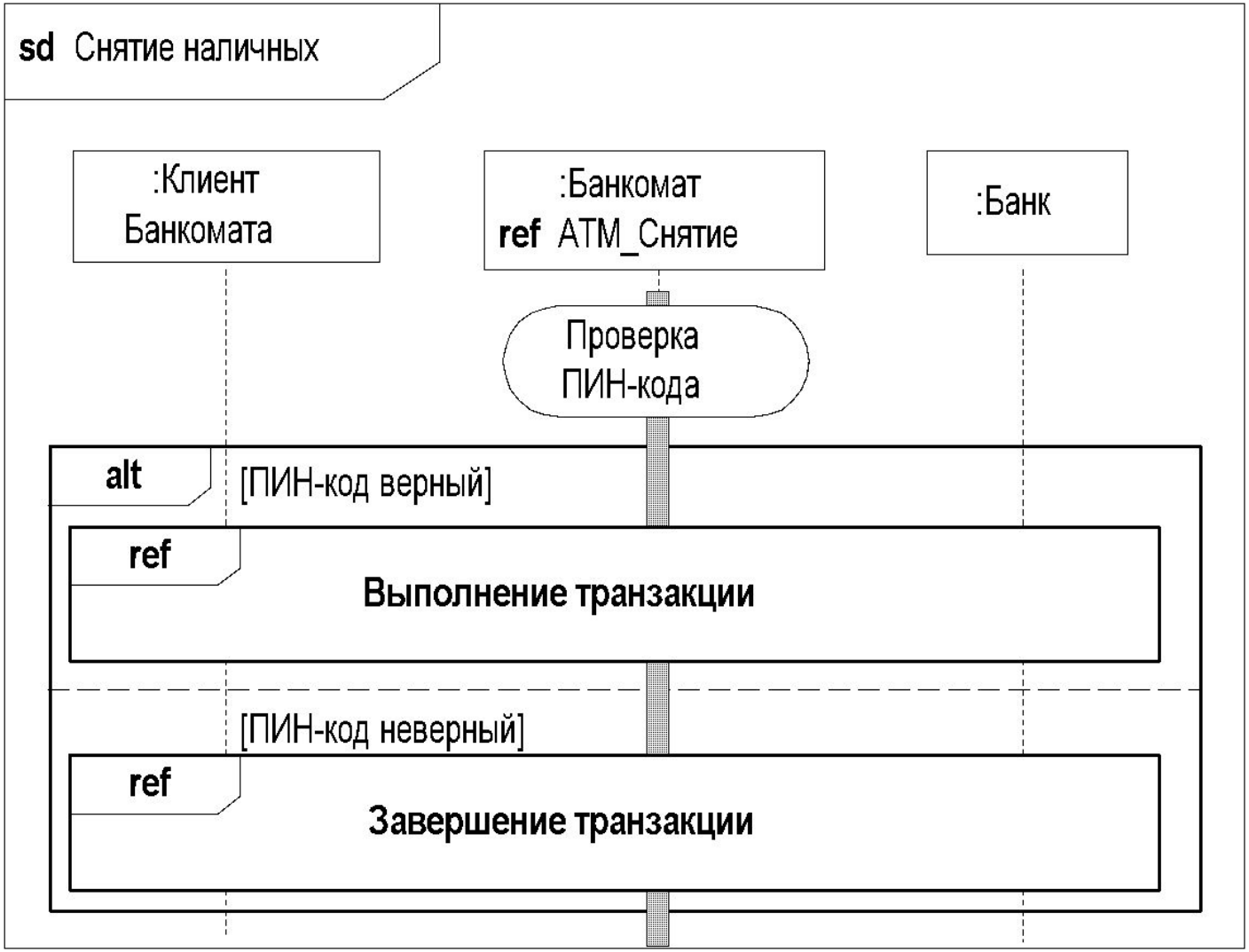


Диаграмма последовательности

Пример представления инварианта состояния в форме ограничения.

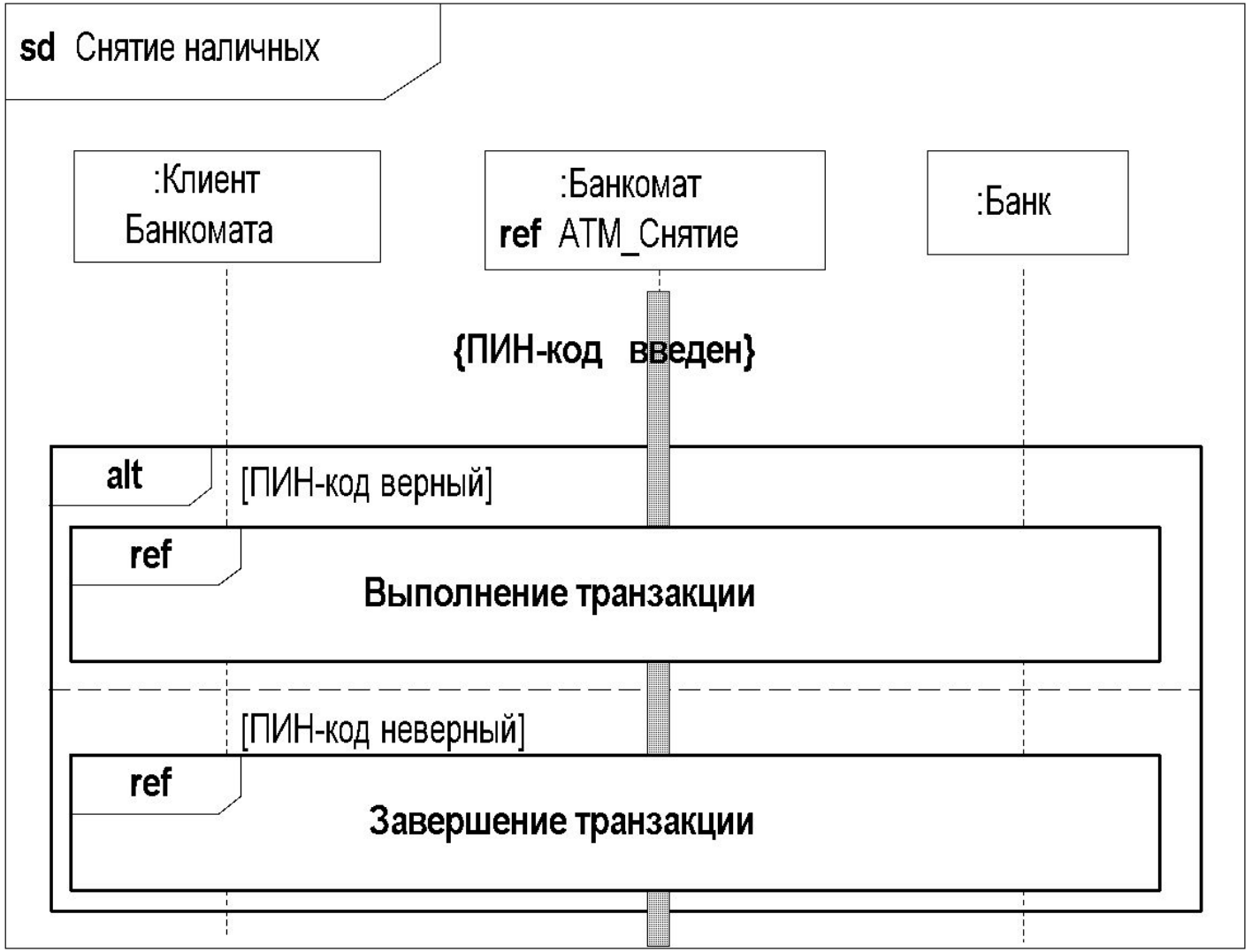


Диаграмма последовательности

Временное ограничение

Time constraint - представляет собой специальное ограничение, записанное в форме временного интервала. Пример:

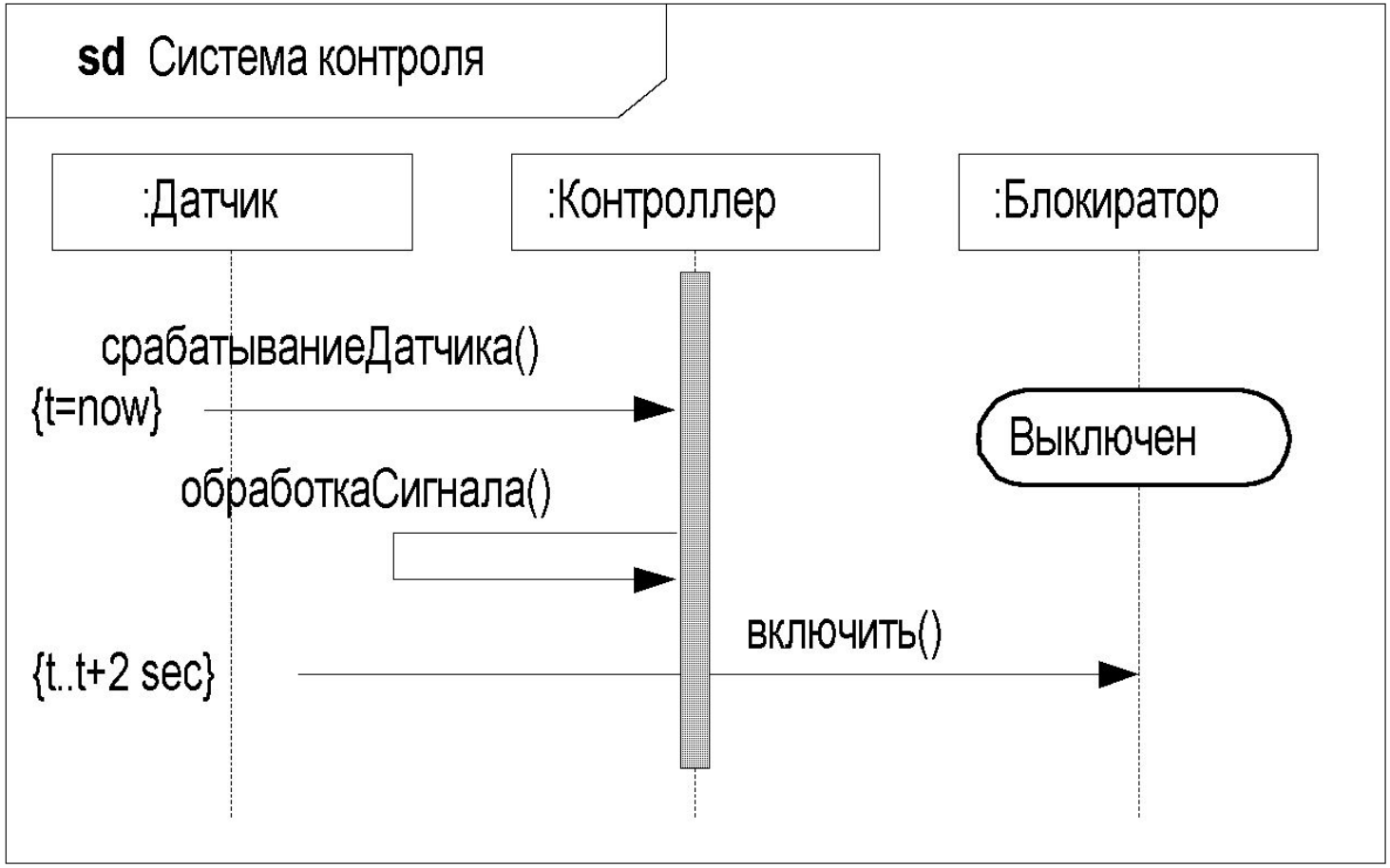


Диаграмма последовательности

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’

Диаграмма последовательности

Ограничение на продолжительность

Duration constraint – определяет ограничение, которое ссылается на некоторый интервал продолжительности. Пример: