

Лабораторная работа № 8

Тема: разработка диаграмм последовательностей.

Цель: изучить принципы построения диаграмм последовательностей языка UML, освоить построение диаграмм последовательностей в CASE-средстве Enterprise Architect 8.

Краткая теория

Диаграмма последовательности – диаграмма, предназначенная для представления взаимодействия между элементами модели программной системы в терминологии линий жизни и сообщений между ними.

Графически диаграмма последовательности имеет два измерения:

- первое – слева направо в виде вертикальных линий каждая из которых соответствует линии жизни отдельного участника взаимодействия;
- второе – вертикальная временная ось, направленная сверху вниз.

Разработка диаграмма последовательности осуществляется в ходе фазы детального проектирования, когда должна быть установлена точная коммуникация внутри процесса в соответствии с формальными протоколами.

В дальнейшем эта диаграмма может быть использована для тестирования приложения или его отдельных модулей.

Взаимодействие – единица поведения некоторого классификатора, которая концентрирует внимание на наблюдаемом обмене информацией между элементами, являющимися участниками этого взаимодействия.

Для моделирования деталей взаимодействия и описания последовательности наступления событий в языке UML 2 используется специальный термин – траектория, которая может быть обозначена в форме:

<наступление события-1, наступление события-2, ..., наступление события-n>

Чередующиеся траектории – объединение двух или более траекторий, так что события из различных траекторий могут происходить в любом порядке в результирующей траектории, в то время как события в одной и той же траектории образуют свой собственный порядок.

Линия жизни представляет одного индивидуального участника взаимодействия или отдельную взаимодействующую сущность. Графическое изображение линий жизни приведено на рисунке 1.

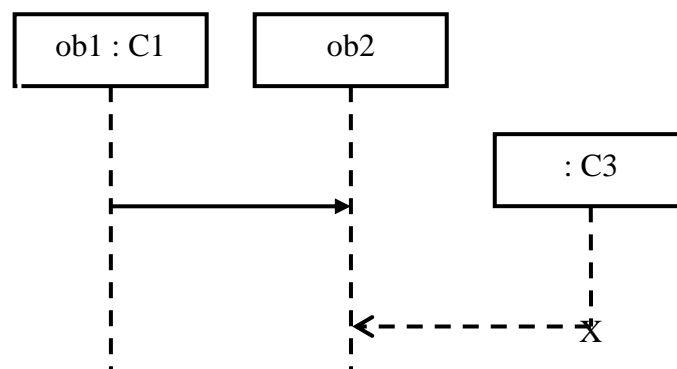


Рисунок 1 – Графическое изображение линий жизни

Информация, идентифицирующая линию жизни, изображается внутри прямоугольника в следующем формате:

```
<идентификатор-линии-жизни> ::=
    ( [<имя-связанного-элемента>[ '[' <селектор> ']' ] ]
      [ : <имя-класса> ] [ декомпозиция ] ) | 'self'
```

где

<селектор> ::= <выражение>

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

Спецификация выполнения предназначена для моделирования состояния активности линии жизни в описываемом взаимодействии. Графическое изображение спецификации приведено на рисунке 2.

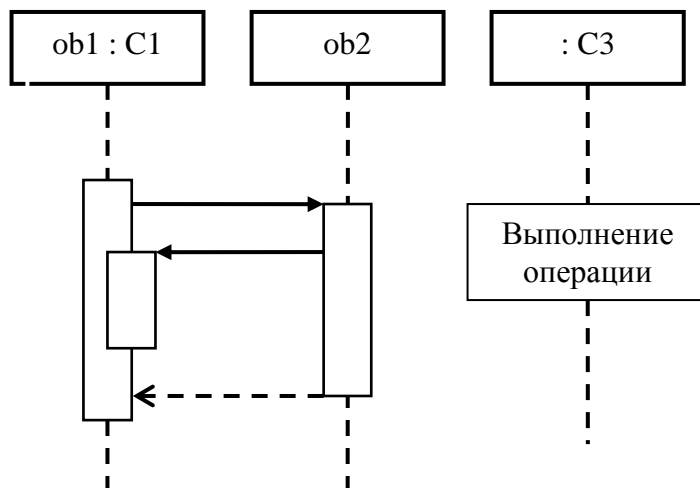







Рисунок 2 – Графическое изображение спецификации выполнения

Сообщение – элемент модели, предназначенный для представления отдельной коммуникации между линиями жизни некоторого взаимодействия. Сорт сообщения представляет собой тип перечисления, который идентифицирует характер коммуникации, лежащей в основе генерации данного сообщения. Сорт сообщения является перечислением следующих значений:

- 'synchCall' – синхронное сообщение 
- 'asynchCall' – асинхронное сообщение 
- 'asynchSignal' – асинхронный сигнал 
- ответное сообщение (reply) 
- сообщение создания объекта (object creation) 

Вид сообщения представляет собой тип перечисления, который идентифицирует тип сообщения. Вид сообщения является перечислением следующих значений:

- 'complete' – полное сообщение, для которого существует событие передачи и приема.
- 'lost' – потерянное сообщение, для которого существует событие передачи и отсутствует событие приема.
- 'found' – найденное сообщение, для которого существует событие приема и отсутствует событие передачи.
- 'unknown' – неизвестное сообщение, для которого отсутствуют событие передачи и событие приема.

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> ] | '-')
```

Примеры:

- отобразить(10, - , 255, 255, «Привет»)
- длина=возвратить() : 96
- рассчитатьНалог(процент=13)

Аргументами могут быть следующие элементы:

- атрибуты передающей линии жизни;
- константы;
- символические значения, которые являются произвольными значениями, представляющими любое допустимое значение;
- явные параметры охватываемого взаимодействия;
- атрибуты класса, владеющего данным взаимодействием.

Виды параллельности вызова операции:

- ‘sequential’ – вызов последовательной операции, для которой не определен механизм управления параллельностью;
- ‘guarded’ – вызов охраняемой операции, для которой могут происходить одновременно несколько вызовов операций у одной линии жизни, но допускается к выполнению только один вызов;
- ‘concurrent’ – вызов параллельной операции, для которой могут происходить одновременно несколько вызовов операций у одной линии жизни, и все они могут быть обработаны параллельно.

Сигнал представляет собой спецификацию асинхронной коммуникации между линиями жизни. Событие сигнала представляет собой прием линией жизни некоторого асинхронного сигнала. Спецификация события сигнала обозначается с использованием следующего формата:

<событие-сигнала> ::= <имя-сигнала> ['(' [<спецификация-назначения>] ')']

где

<спецификация-назначения> ::= <имя-атрибута> [',' <имя-атрибута>]*

Примеры:

- mySignal(s: String)
- оповеститьКлиента

Фрагмент взаимодействия является абстрактным понятием для представления общей единицы или части взаимодействия. Комбинированный фрагмент – элемент модели, предназначенный для представления внутренней логической структуры фрагментов взаимодействия. Операнд взаимодействия – отдельный фрагмент взаимодействия, предназначенный для использования в качестве внутренней части комбинированного фрагмента. Виды операторов: alt, assert, break, critical, ignore, consider, loop, neg, opt, par, seq, strict.

Ограничение взаимодействия представляет собой логическое выражение, которое выступает в роли сторожевого условия некоторого операнда в комбинированном фрагменте.

<ограничение-взаимодействия> ::=
['[' (<логическое выражение> | 'else') ']']

Оператор взаимодействия **alt** специфицирует комбинированный фрагмент *альтернативы*, который представляет некоторый выбор поведения (рисунок 3).

Оператор взаимодействия **assert** специфицирует комбинированный фрагмент *утверждения*, который представляет некоторое утверждение. В этом случае единственными следствиями, которые имеют возможность продолжения, являются сообщения или вложенные фреймы данного операнда. Комбинированный фрагмент **assert** часто объединяется с операторами **ignore** или **consider**.

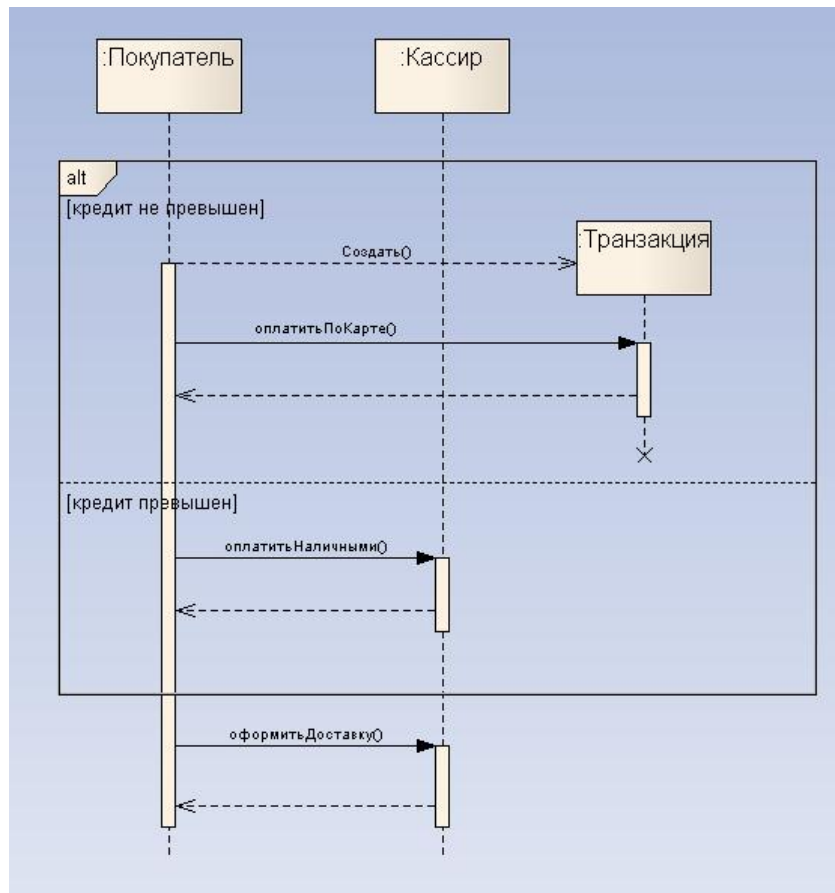


Рисунок 3 – Оператор взаимодействия alt

Оператор завершения **break** специфицирует комбинированный фрагмент *завершение*, который представляет некоторый сценарий завершения (рисунок 4).

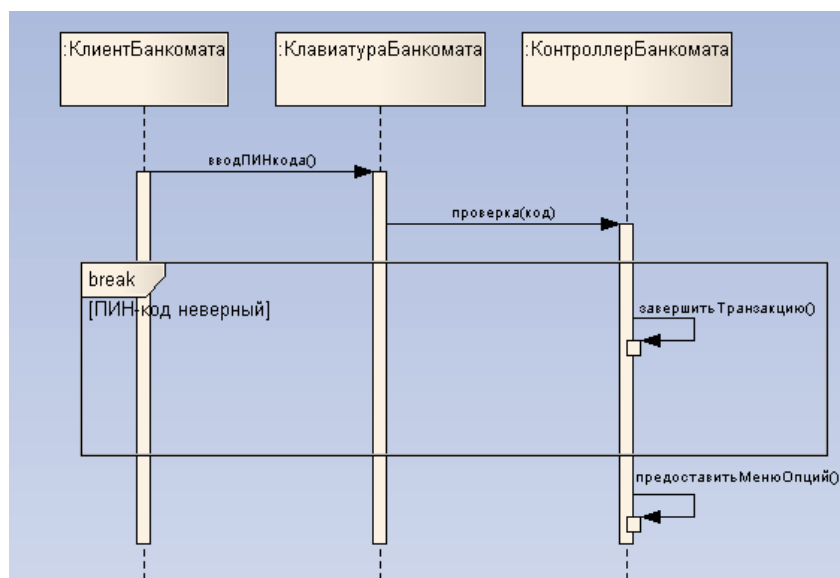


Рисунок 4 – Оператор взаимодействия break

Оператор взаимодействия **critical** специфицирует комбинированный фрагмент *критический регион*, траектории которого не могут чередоваться с другими спецификациями наступления событий на тех линиях жизни которые этот регион покрывает (рисунок 5).

Оператор взаимодействия **par** специфицирует комбинированный фрагмент *параллельный*, который представляет некоторое параллельное выполнение взаимодействий своих операндов (рисунок 5).

Оператор взаимодействия **consider** специфицирует комбинированный фрагмент *рассмотрение*, в котором изображены только те типы сообщений, какие должны рассматриваться в этом фрагменте. Синтаксис:

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

Оператор взаимодействия **ignore** специфицирует комбинированный фрагмент *игнорирование*, в котором имеются некоторые типы сообщений, не изображенные на этой диаграмме (рисунок 6). Синтаксис:

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

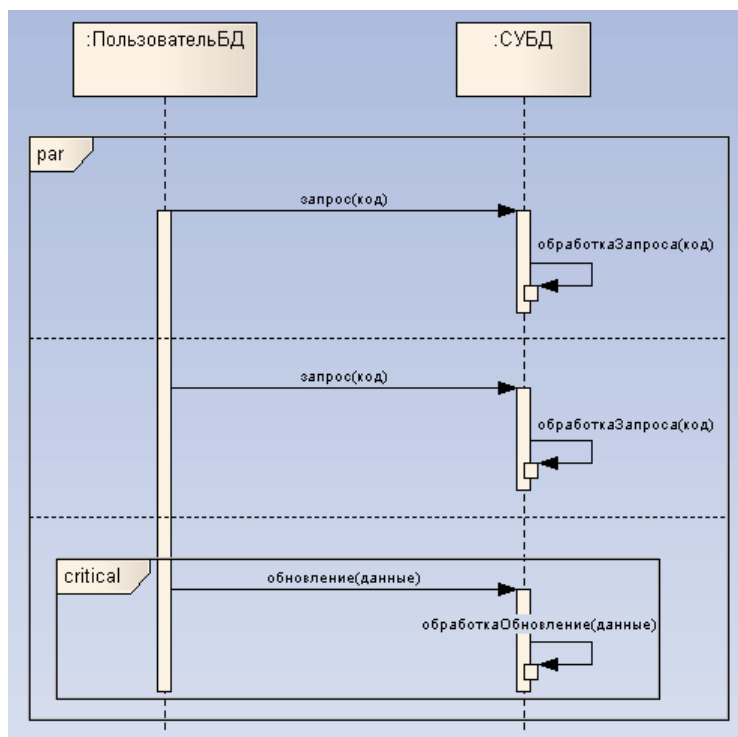


Рисунок 5 – Операторы взаимодействия par и critical

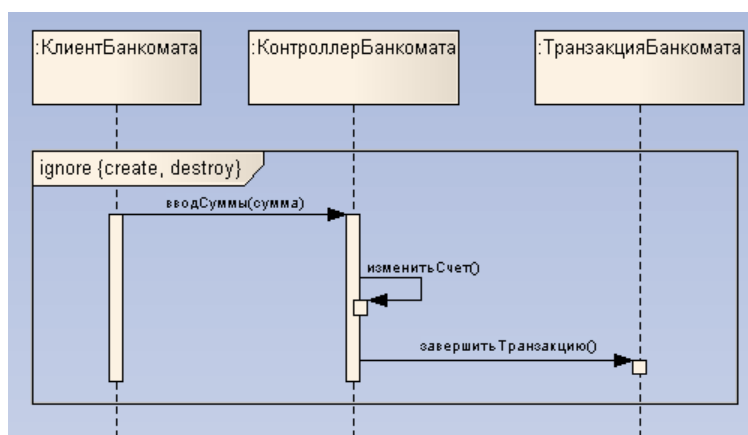


Рисунок 6 – Оператор взаимодействия ignore

Оператор взаимодействия **loop** специфицирует комбинированный фрагмент *цикл*, который представляет собой циклическое повторение некоторой последовательности сообщений (рисунок 7). Синтаксис:

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

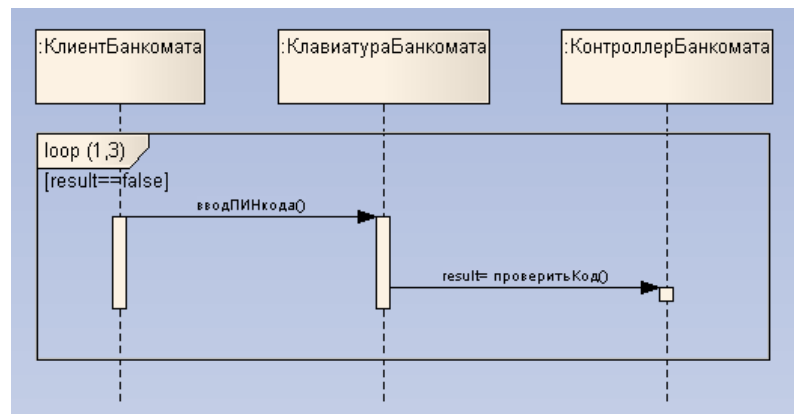


Рисунок 7 – Оператор взаимодействия loop

Оператор взаимодействия **neg** специфицирует комбинированный фрагмент *отрицание*, представляющий траектории, которые определяются как недействительные или недопустимые (рисунок 8).

Оператор взаимодействия **opt** специфицирует комбинированный фрагмент *необязательный*, который представляет выбор поведения, когда либо выполняется единственный операнд, либо вовсе ничего не выполняется (рисунок 8).

Оператор взаимодействия **seq** специфицирует комбинированный фрагмент *слабое следование*, 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 разных операндов могут происходить в произвольном порядке;
- наступление событий на одной линии жизни у различных операндов упорядочиваются сверху вниз.

Оператор взаимодействия **strict** специфицирует комбинированный фрагмент *строгое следование*, который состоит из нескольких операндов и представляет строгий порядок следований поведения отдельных операндов.

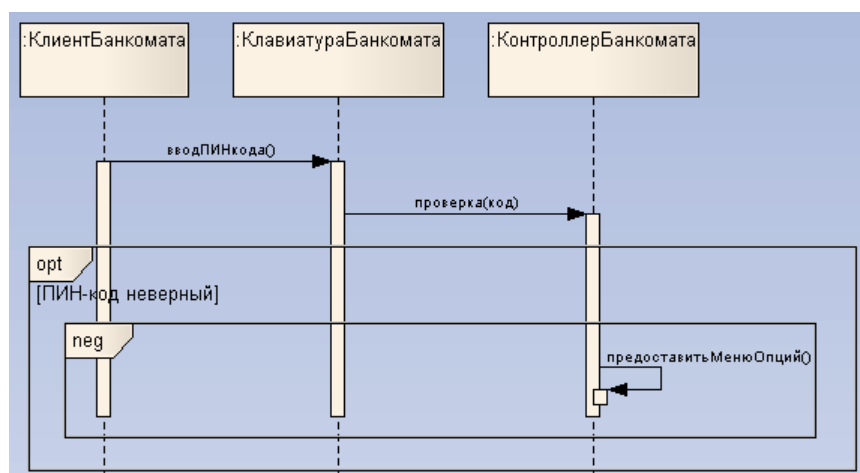


Рисунок 8 – Операторы взаимодействия opt и neg

Использование взаимодействия – элемент модели, представляющий параметризованную ссылку на некоторое взаимодействие в контексте другого взаимодействия (рисунок 9). Синтаксис:

```

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

```

[':' <возвращаемое-значение>
 <io-аргумент> ::= <in-аргумент> | 'out' <out-аргумент>

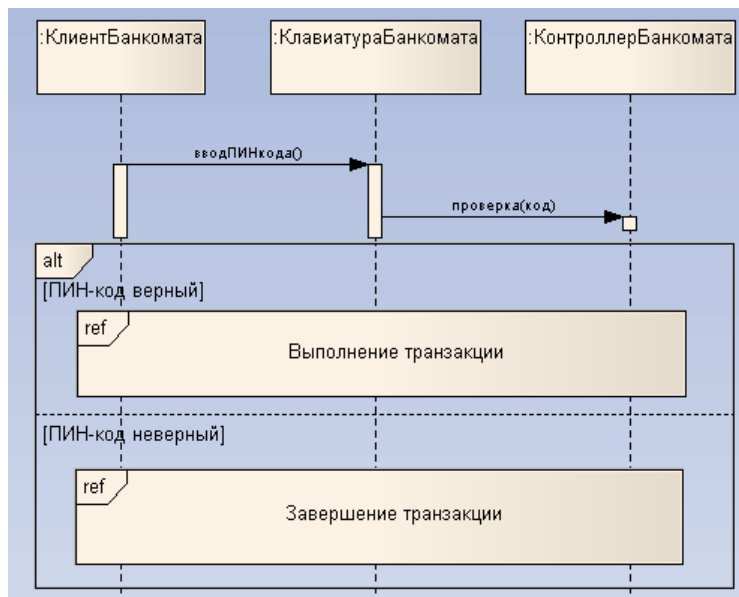


Рисунок 9 – Оператор взаимодействия ref

Ход работы

В рамках выполнения данной лабораторной работы необходимо на разработанной ранее диаграмме вариантов использования выбрать вариант использования, который включает (отношение «include») или может быть расширен (отношение «extend») другим вариантом использования.

Для обоих вариантов использования разработать диаграммы последовательностей, соблюдая следующие условия:

- диаграмма последовательностей для дополнительного варианта использования должна включаться в диаграмму для основного варианта использования с помощью оператора взаимодействия ref;
- на диаграммах должны использоваться не менее трех комбинированных фрагментов с различными операторами (alt, loop, break и т.д.);
- линии жизни и передаваемые или принимаемые ими сообщения должны соответствовать классам и их характеристикам, определенным в рамках лабораторной работы № 7.