Example: Interaction Diagrams

Sequence Diagram

Communication Diagram
Example Sequence Diagram
Creation and deletion of participants
Synchronous– vs Asynchronous Calls:

**Synchronous**

\[
\begin{align*}
  b &. m(4); \\
  c &. n(...) \quad // \text{Starts after } m \text{ has returned}
\end{align*}
\]

**Diagram:**

- **Synchronous calls**
  - a:A
  - b:B
  - c:C
  - m(4) starts before n(...)
Synchronous– vs Asynchronous Calls:

**Synchronous**

\[ b.m(4); \]
\[ c.n(...) \] // Starts after m has returned

**Asynchronous**

// (new Thread(){ public void run() {b.m(4);}}).start();
new Thread(() -> {b.m(4);}).start(); // Using Lambdas from Java 8
\[ c.n(...) \] // Starts immediately after m has been called
ATM to Bank: Synchronous Version

1. User enters PIN
2. Bank verifies PIN
3. True
4. Bank dispenses money
5. Transaction complete
Realisation with Interaction Frames
## Interaction Frame Operators I

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>alt</code></td>
<td>Alternative multiple fragments; only the one whose condition is true will execute (Figure 4.4).</td>
</tr>
<tr>
<td><code>opt</code></td>
<td>Optional; the fragment executes only if the supplied condition is true. Equivalent to an <code>alt</code> with only one trace (Figure 4.4).</td>
</tr>
<tr>
<td><code>par</code></td>
<td>Parallel; each fragment is run in parallel.</td>
</tr>
<tr>
<td><code>loop</code></td>
<td>Loop; the fragment may execute multiple times, and the guard indicates the basis of iteration (Figure 4.4).</td>
</tr>
<tr>
<td><code>region</code></td>
<td>Critical region; the fragment can have only one thread executing it at once.</td>
</tr>
<tr>
<td><code>critical</code></td>
<td></td>
</tr>
<tr>
<td><code>neg</code></td>
<td>Negative; the fragment shows an invalid interaction.</td>
</tr>
<tr>
<td><code>ref</code></td>
<td>Reference; refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in the interaction. You can define parameters and a return value.</td>
</tr>
<tr>
<td><code>sd</code></td>
<td>Sequence diagram; used to surround an entire sequence diagram, if you wish.</td>
</tr>
</tbody>
</table>
Nested sequence diagrams
Usages of sequence diagrams

- Abstract: show the execution (i.e. exchange of messages) of a system
- Concrete
  - Design (c.f. CRC cards)
  - Visualize program behaviour
  - Visualize model execution → use case realization
Use Case Name: Check out

Summary: Checks out a book from the library

Actors: User

Preconditions: true

Basic course of events

1. User scans his library card
2. User repeats
   2.1 select check out
   2.2 scan the book
   2.3 System confirms loan

Alternative paths

Postconditions

Book is loaned by the user
Use Case scenarios as Sequence Diagrams

Main scenario

```
Main scenario

sd: borrow book success

User
  scanLibraryCard(bor)
    true
  checkOut()
  scanBook(b)
    true
  loop

Library
```
System design for the main use case scenario

sd: borrow book success

User

Library

bor:Borrower

b1:Book

b2:Book

scan library card(bor)

true

canBorrow()

loop

checkOut()

[ b in bor.books ]

loop

isOverdue()

false

true

true

true

checkout(bor)