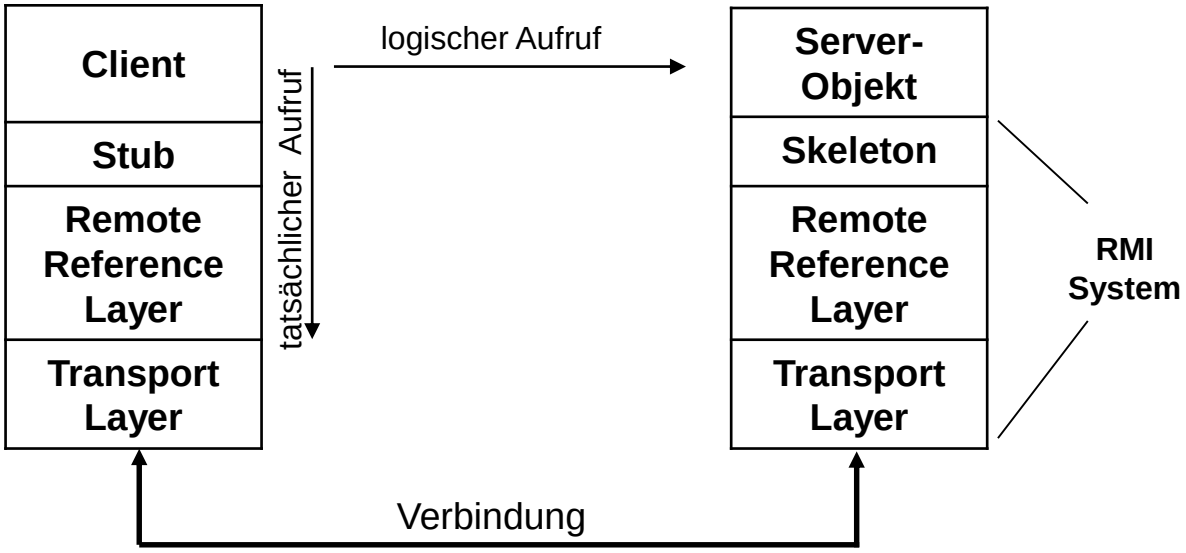
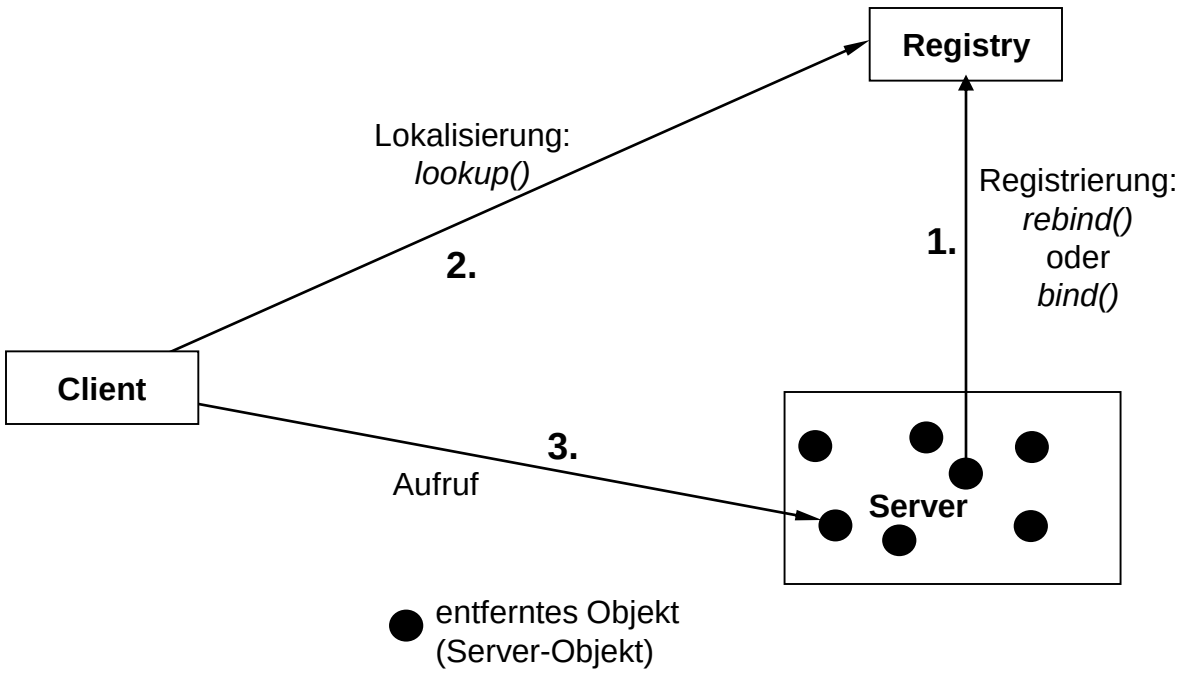
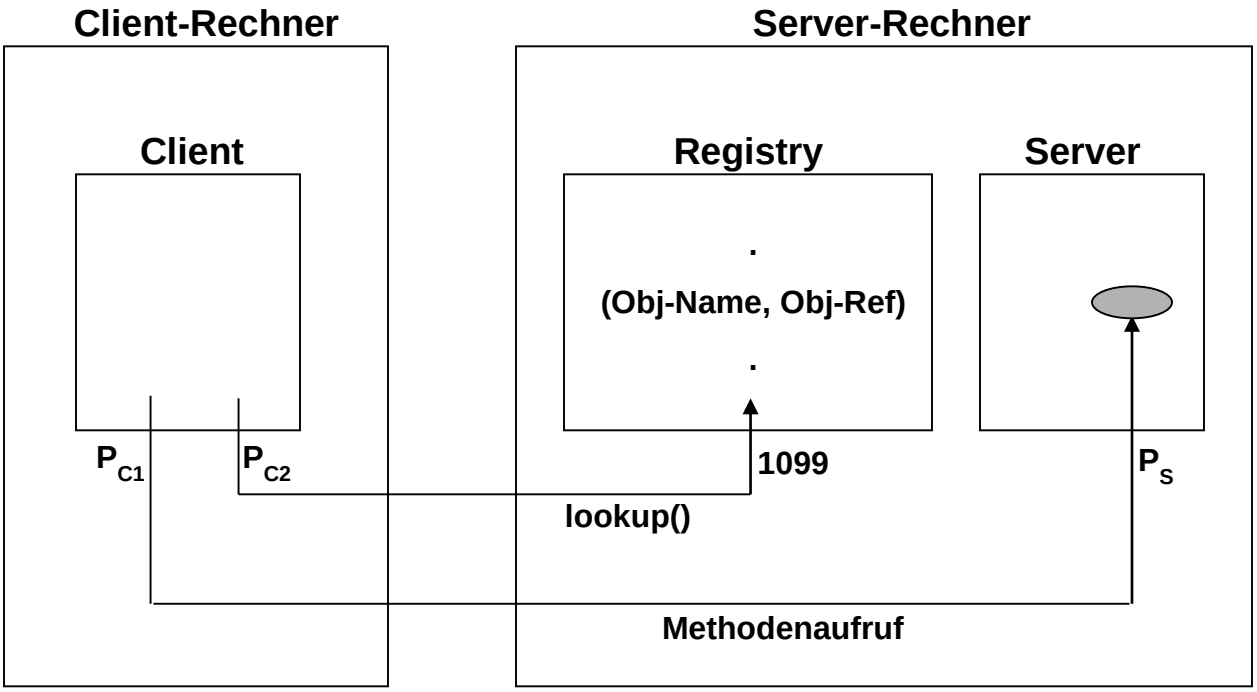




RMI-Kommunikationsarchitektur und -Ablaufmodell



P_{C1}, P_{C2}, P_S, 1099: Portnummern

Obj-Name: Name des entfernten Objektes

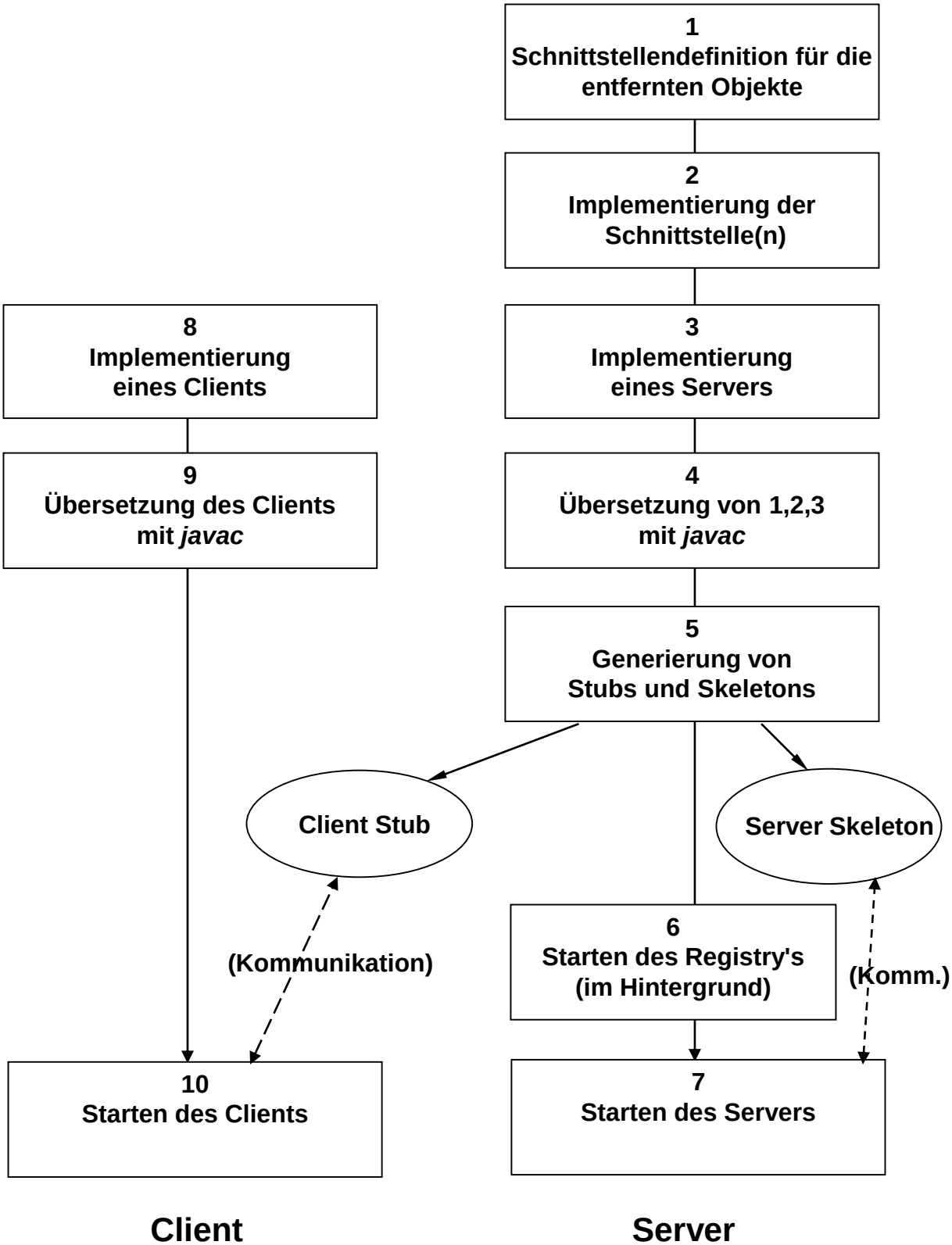
Obj-Ref: Referenz des entfernten Objektes mit:

- Internet-Adresse des Server-Rechners,
- Portnummer des entfernten Objektes,
- Identifikator des entfernten Objektes

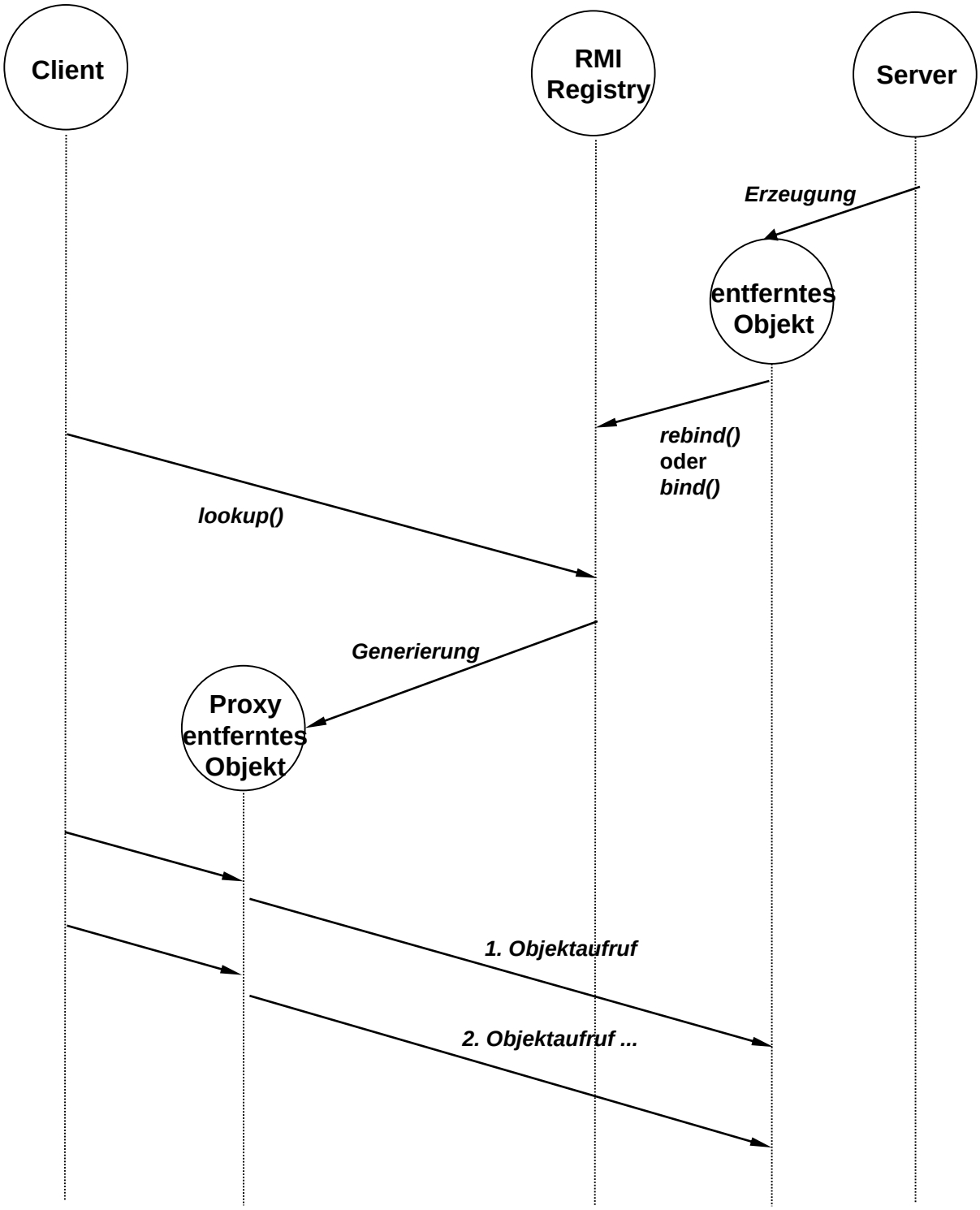
entferntes Objekt (Server-Objekt)

Registry und Server-Objekt auf demselben Server-Rechner!

RMI-Kommunikation



RMI-Entwicklungsprozess



RMI-Szenario

```
// Schnittstellendefinition: Datenstruktur
import java.io.*;

public class Complex implements Serializable {
    public static final long serialVersionUID = 1L;
    public int real;
    public int imag;
    public Complex () { this.real = 0; this.imag = 0; }
    public Complex (int r, int i) {this.real = r; this.imag = i;}
}
```

```
// Schnittstellendefinition: Exception
import java.rmi.*;

public class ComplexException extends Exception {
    public ComplexException (String msg) {super(msg);}
}
```

```
// Schnittstellendefinition: Anwendungsmethode
import java.rmi.*;

public interface ComplexAdder extends Remote {
        Complex add (Complex a, Complex b)
                throws RemoteException, ComplexException;
}
```

RMI: Schnittstellen-Definition

```
// Schnittstellenimplementierung
import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;

public class ComplexAdderImpl
      extends UnicastRemoteObject implements ComplexAdder {

   private static final long serialVersionUID = 1L;
   private Complex zahl;

   public ComplexAdderImpl (String name, Registry reg)
                                                   throws RemoteException {
      zahl = new Complex();
      try {
         reg.rebind (name, this);
      }
      catch (Exception e) {
         System.out.println ("Exception: " + e.getMessage());
      }
   }

   public synchronized Complex add (Complex a, Complex b)
            throws RemoteException, ComplexException {
      zahl.real = a.real + b.real;
      zahl.imag = a.imag + b.imag;
      if (zahl.real == 0 && zahl.imag == 0)
            throw new ComplexException ("Ergebnis ist Null.");
      return zahl;
   }
}
```

```
// Server

// Usage: java ComplexAdderServer

import java.rmi.*
import java.rmi.registry.*

public class ComplexAdderServer {

    public static void main (String args[]) {

        try {
            Registry reg = LocateRegistry.createRegistry
                                        (Registry.REGISTRY_PORT);
            ComplexAdderImpl ad =
                new ComplexAdderImpl ("myComplexAdder", reg);
            System.out.println ("ComplexAdder Server ready.");
        }
        catch (Exception e) {
            System.out.println ("Exception: " + e.getMessage());
        }
    }
}


// vollständiger Konstruktoraufruf, Beispiel:

// ComplexAdderImpl ad =

//     new ComplexAdderImpl ("rmi://lux3:1099/myComplexAdder", reg);


// in allen RMI-spezifischen Methoden (bind(), rebind(), lookup(), list(), ...)
// gelten die Defaults localhost für die Server-Maschine
// und REGISTRY_PORT = 1099 für den Port
```

RMI-Beispiel: Server

```
// Client
// Usage: java ComplexAdderClient <hostname> <zahl> <zahl> <zahl> <zahl>

import java.rmi.*;
import java.rmi.registry.*;

public class ComplexAdderClient {
      public static void main (String args[]) {
            try {
               Registry reg =
                  LocateRegistry.getRegistry(Registry.REGISTRY_PORT);
               ComplexAdder ad =
                        (ComplexAdder) reg.lookup ("myComplexAdder");
               Complex s1   = new Complex();
               Complex s2   = new Complex();
               Complex s    = new Complex();
               s1.real    = Integer.parseInt(args[1]);
               s1.imag  = Integer.parseInt(args[2]);
               s2.real    = Integer.parseInt(args[3]);
               s2.imag  = Integer.parseInt(args[4]);
               s = ad.add(s1,s2);
               System.out.println ("Summe: (" + s.real + "," + s.imag + ")");
            }
            catch (Exception e) {
               System.out.println ("Exception: " + e.getMessage());
            }
      }
}

// vollständiger lookup-Aufruf:
// ... = (ComplexAdder) reg.lookup
//               ("rmi://" + args[0] +  ":1099/" + "myComplexAdder");

// in allen RMI-spezifischen Methoden (bind(), rebind(), lookup(), list(), ...)
// gelten die Defaults localhost für die Server-Maschine
// und REGISTRY_PORT = 1099 für den Port
```


RMI-Beispiel: Client

```
javac ComplexAdder.java
javac ComplexAdderImpl.java
javac ComplexAdderServer.java
javac ComplexAdderClient.java


java ComplexAdderServer
// auf Server-Maschine

java ComplexAdderClient
      <hostname der Server-Maschine>
                        <zahl> <zahl> <zahl> <zahl>
// auf Client-Maschine
```

RMI-Beispiel: Entwicklung