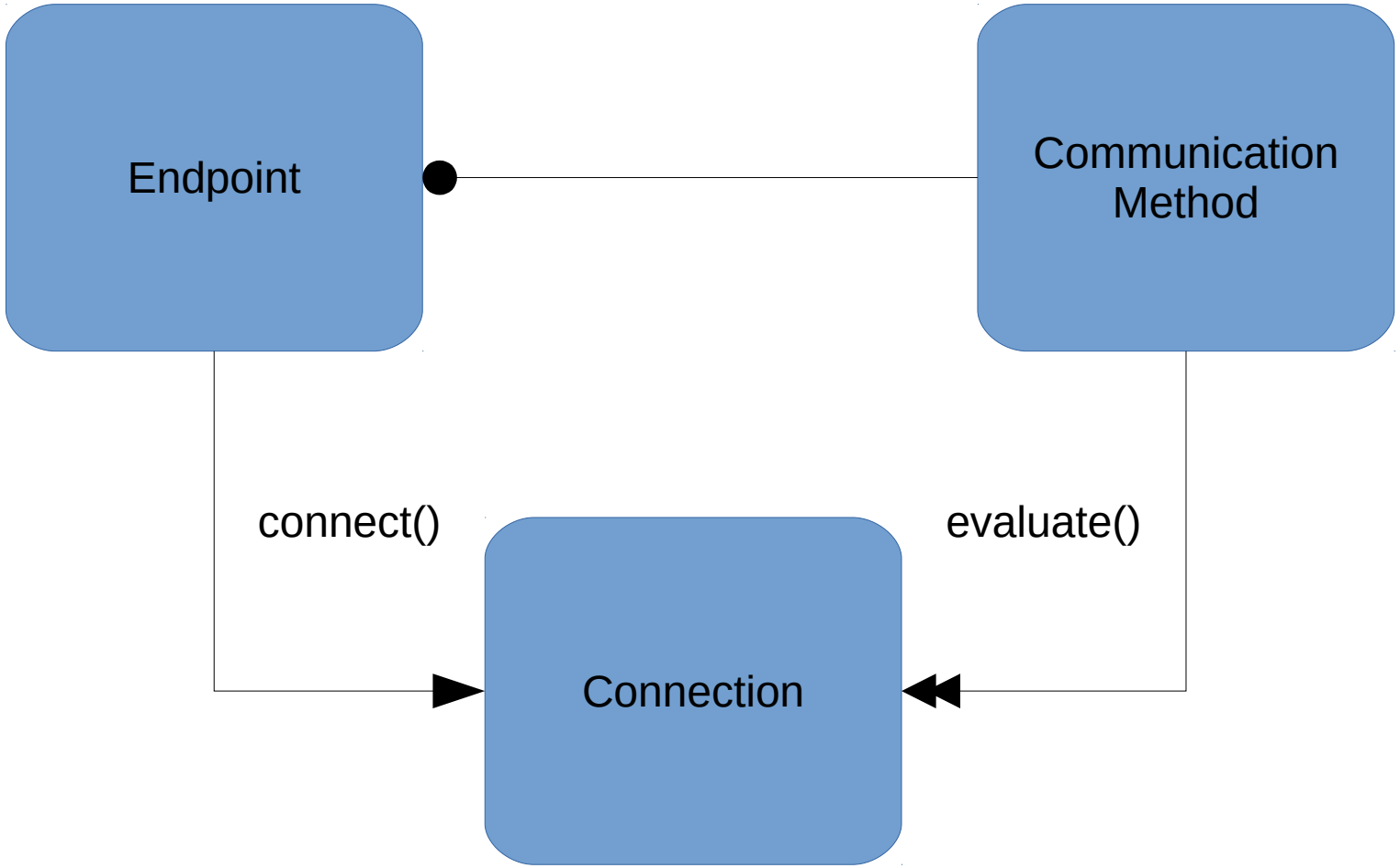
Kommunikations-Framework

Ziel: Abstraktion der Kommunikation mit anderen Software-Systemen

- einfache Nutzung für alle Entwickler
- weitgehend automatisches Resource-Handling
- zentrale Stelle für Logging, Konfiguration etc.
- Kommunikationsobjekte als Parameter
- basierend auf Functions für einfache Nutzung bestehender Funktionalitäten

Klassenstruktur



Funktionen als Basis

```
public abstract class Endpoint
   extends AbstractFunction<Relatable, Connection>
{
   public Connection connect(Relatable rParams) {}
}

public abstract class CommunicationMethod<I, O>
   extends AbstractBinaryFunction<I, Connection, O>
{
   public abstract O doOn(Connection rConnection, I rInput);
   public O getFrom(Connection rConnection, I rInput) {}
}
```

Beispiel: statische Nutzung

```
Endpoint aExampleOrg = Endpoint.at("http://example.org");

Function<String, String> fGetBody =
   httpGet("index.html").from(aExampleOrg)
                        .then(find("(?s)<body>(.*)</body>"));

String sBody = fGetBody.result();
```

Beispiel: dynamische Nutzung

```
Endpoint aExampleOrg = Endpoint.at("http://example.org");

try (Connection rConnection = aExampleOrg.connect())
{
   Function<String, String> fGetBody =
      httpGet().then(find("(?s)<body>(.*)</body>"));

   String sBody = fGetBody.getFrom(rConnection,
                                   "index.html");
}
```

Beispiel: Connection-Parameter

```
Relatable aParams = new RelatedObject();

aParams.set(CommunicationRelationTypes.USER_NAME, "foo");
aParams.set(CommunicationRelationTypes.PASSWORD, "bar");

EndpointChain<String, List<String>> fListRoot =
   SftpEndpoint
      .listDir(".")
      .from(Endpoint.at("sftp://docker1.lemarit.com:2223"));

List<String> aRootFiles = fListRoot.with(aParams);
```