Verteilte Software

Galina Rudolf
Galina.Rudolf@informatik.tu-freiberg.de

Erste Vorlesung: 24.10.22 11:30 Uhr HUM-1115
Erste Übung: 1.11.22 7:30 Uhr RAM-2119
Unterlagen: OPAL

Quelle: https://www.spektrum.de

Ziel: Verteilte Anwendungen erstellen zu können, unter Verwendung verschiedener (Java) - Technologien

Prüfung: mündlich mit einem schriftlichen Anteil, 30 min Vorbereitung, 30 min Diskussion

Was ist verteilte Software? Was ist verteilte Anwendung?

Verteilte Software = verteilte Anwendung

eine komplexe Anwendung, die in einem **verteilten System** abläuft

„Ein verteiltes System ist eine Sammlung unabhängiger Computer, die den Benutzern als ein einziges zusammenhängendes System erscheint.“
Andrew Tanenbaum

Verteiltes System besteht aus mehreren Prozessen, die u.U. auf verschieden Rechnern laufen.

Prozesse erbringen gemeinsam die Leistung zur Lösung einer Aufgabe
Zum Austausch von Informationen müssen Prozesse **miteinander interagieren**.

Beispiele?

Muss es immer eine verteilte Anwendung sein?

Vorteile:

effiziente Nutzung von Ressourcen

betriebswirtschaftliche Gründe (Hochleistungsrechner vs. mehrere PCs)

Zugriff auf entfernte Daten (Ressourcen) möglich

robust durch redundante Datenkopien

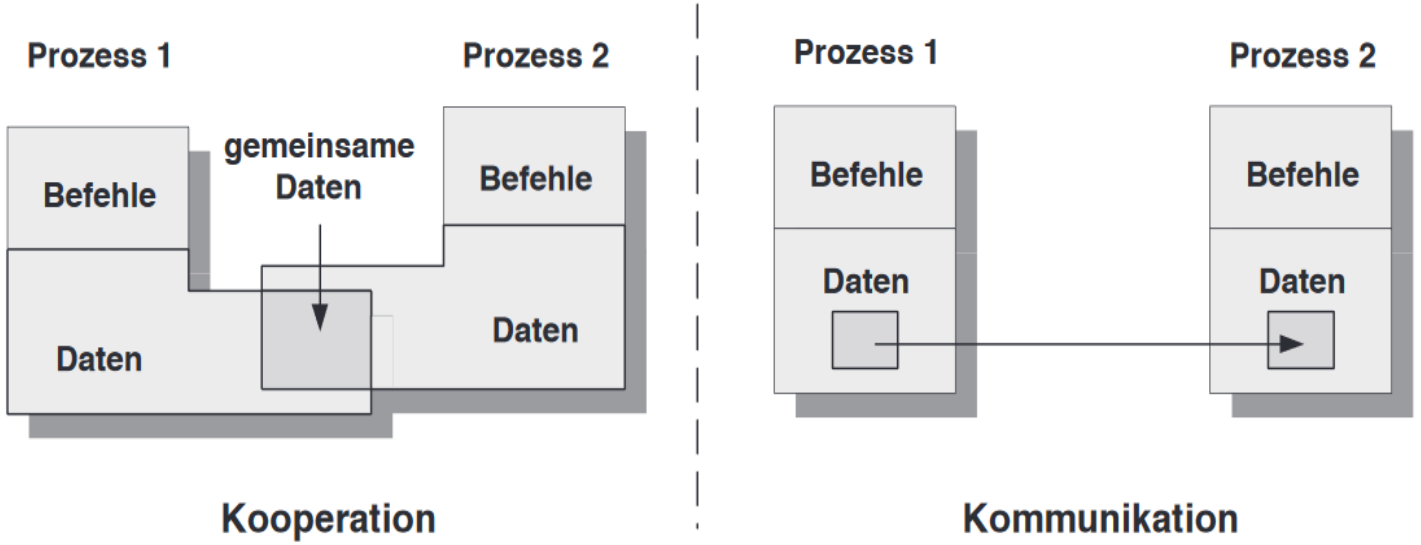
Nachteile:

erhöhte Komplexität in allen SWE-Phasen:

- Entwurf: Modell muss Kommunikationsmechanismen beachten
- Implementierung: durch Kommunikation zusätzliche Fehlerfälle
- Test: entfernte Partner müssen simuliert werden
- Betrieb: Ausfälle einzelner Rechner können zu Gesamtausfällen führen
- Versionisierung und Konfiguration: muss konform gehen, um Austausch von Teilen zu ermöglichen

Interaktion
Interaktionsmuster

gemeinsamer Datenbereich **Austausch über Nachrichten**



Quelle:
Oliver Haase: Kommunikation in verteilten Anwendungen, Oldenburg

Verteilung

Was wird verteilt?

- Daten
- Ausführung der Aufgaben

Sichtweise:

physisch (systemorientiert) – Verteilung der Hardware bezieht sich auf das System unabhängiger entfernter Rechner, Verteilung der Software wird in der Verteilung auf Prozesse (Threads) abgebildet

logisch (problemorientiert) - Verteilung auf Klassen (Objekte, Interfaces), Pakete

Eigenschaften einer verteilten Anwendung

• Heterogenität

Anwendung ist eine Gesamtheit, die aus nicht gleichartigen Elementen besteht

- einheitliches Programmiermodell zur Entwicklung verteilter Software
  Prozedurfernaufruf (remote procedure call, RPC)
  Objektfernaufruf (remote method invocation, RMI)

- grundlegende Bausteine einer Anwendung bilden Prozesse (Threads) und Nachrichtenaustausch

- hardwareunabhängiger übertragbarer Maschinencode inkl. Interpreter
  Zwischenkode und virtuelle Maschine

• Nebenläufigkeit

Die Fähigkeit einer Anwendung mehrere Aufgaben gleichzeitig ausführen zu können
Nebenläufiger Server bedient mehrere Klienten gleichzeitig

- **Fähigkeit zur Fehlerbehandlung**

Die Fähigkeit die Fehler zu erkennen und darauf zu reagieren

Fehler können erkennbar (z.B. durch Prüfsummen) und nicht erkennbar sein (z.B. ein Server-Ausfall). Die Herausforderung ist, mit nicht erkennbaren Fehlern umzugehen.

- Fehler verbergen (maskieren): z.B. verlorene Nachrichten wiederholen, Dateien auf mehrere Datenträger sichern) oder abschwächen (z.B. fehlerhafte Nachrichten verwerfen).
- Fehler tolerieren: einige Fehler können hingenommen werden und ggf. dem Anwender gemeldet werden.

- **Sicherheit**

Informationen sicher über ein Netzwerk zu übertragen

- Vertraulichkeit: Informationen (Inhalte inkl. Absenderidentität) vor Unbefugten schützen (Verschlüsselung)
- Integrität: Schutz vor Veränderung oder Beschädigung der Informationen
- Verfügbarkeit: Vermeidung der Störungen während des Betriebs

- **Verteilungstransparenz**
  - Ziel jeder verteilten Anwendung
  - transparent heißt
    die Verteilung ist für Nutzer nicht sichtbar

Zugriffstransparenz (wie die Daten, in welchem Format gespeichert sind)
Ortstransparenz (wo die Ressourcen sich befinden)
Migrationstransparenz (pre-Session Mobilität)
Relokationstransparenz (mid-Session Mobilität)
Persistenztransparenz (flüchtige oder persistente Speicherung)
Nebenläufigkeitstransparenz (mehrere Prozesse)
Replikationstransparenz (mehrere Kopien)
Fehlertransparenz (Verbergen von Ausfällen von Teilsystemen)

- **Skalierbarkeit**

skalierbar heißt kann wachsen ohne Veränderung der Eigenschaften

Größe:

Möglichkeit des Hinzufügens von Komponenten

geographische Verteilung:

System funktioniert unabhängig von der Entfernung

administrative Verteilung:

System funktioniert über die Grenze der administrativen Domäne hinweg (verschiedene Sicherheitsrichtlinien)

Architektur (Struktur des Gesamtsystems)

„Architektur ist eine strukturierende oder hierarchische Anordnung der Systemkomponenten sowie Beschreibung ihrer Beziehungen.“
Helmut Balzert

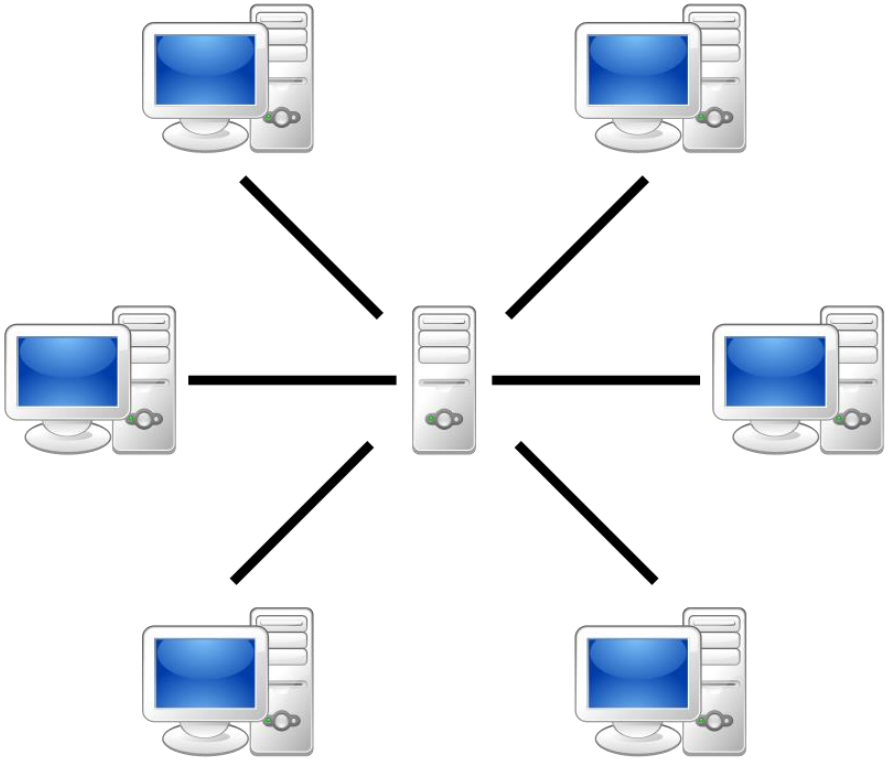
Wie wird verteilt?

- statisch festgelegte Architektur:
  Client-Server, Peer-to-Peer

- In Rahmen einer konkreten Kommunikation:
  Teilnehmer können unterschiedliche Rollen annehmen

Architektur

Client – Server



Jeder Rechner im Kontext eines Dienstes ist entweder Client oder Server

Client: nutzt Dienst

Server: bietet Dienst an

Basis für die Kommunikation: Protokoll

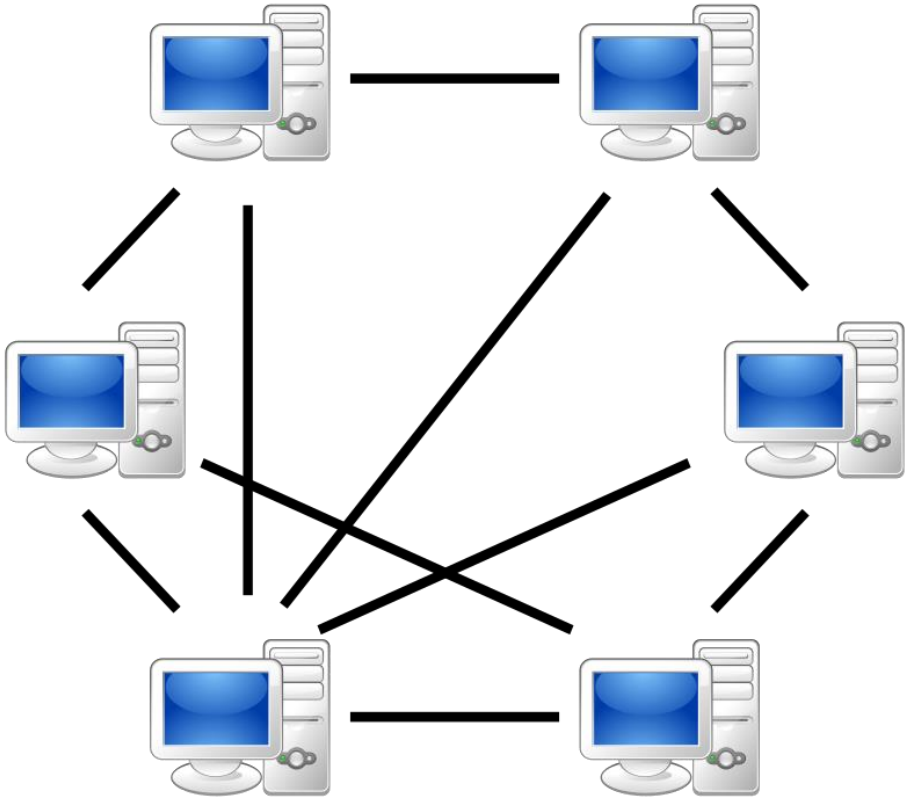
Ein Server kann auf mehrere Rechner verteilt oder repliziert werden

Ein Server kann andere Server beanspruchen

Quelle:
Von User:Mauro Bieg, https://commons.wikimedia.org/w/index.php?curid=2551745

Architektur

Peer – To – Peer



Peer engl.: Gleichgestellter

nutzt oder bietet Dienste an im Kontext einer konkreten Kommunikation

ohne zentrale Serverkomponente, aber meistens wird ein Rendezvous - Server benötigt

skalierbar bezüglich der Größe

hoch dynamisch

Quelle:
Von User:Mauro Bieg, https://commons.wikimedia.org/w/index.php?curid=2551723

Kommunikationsparadigmen

Nachrichtenbasierte Kommunikation

über Dienstprimitiven: send und receive,
verbindungsorientiert, paketorientiert
Sockets, JMS (Java Message Service)

Fernaufruf

Aufruf von Funktionen (Prozeduren) im anderen Adressraum
(Remote Procedure Call - RPC)

Implementierungen:

Methodenfernaufruf (Remove Method Invocation - RMI)
CORBA (Common Object Request Broker Architecture)
Distributed Component Object Model (DCOM)

Kurz Java-Grundlagen (OO und nicht OO-Konzepte)
Ein- und Ausgabe (Streams u.a.)
Nebenläufigkeit (Threads, Runnable, ThreadPools etc.)
Sockets
Android (GUI, Componenten, Hintergrundausführung)
RMI (Remote Method Invocation)
Jakarta Messaging (Java Message Service - JMS)
Servlets im Kontext von HTML, Sicherheit in Web-Anwendungen

Java – Programmiersprache und Java - Technologie

1990 - erste Arbeiten bei Sun Microsystems (jetzt Oracle) unter Codenamen „Oak“ (Eiche)
Zielstellung: eine Programmierplattform für Haushaltelektronik

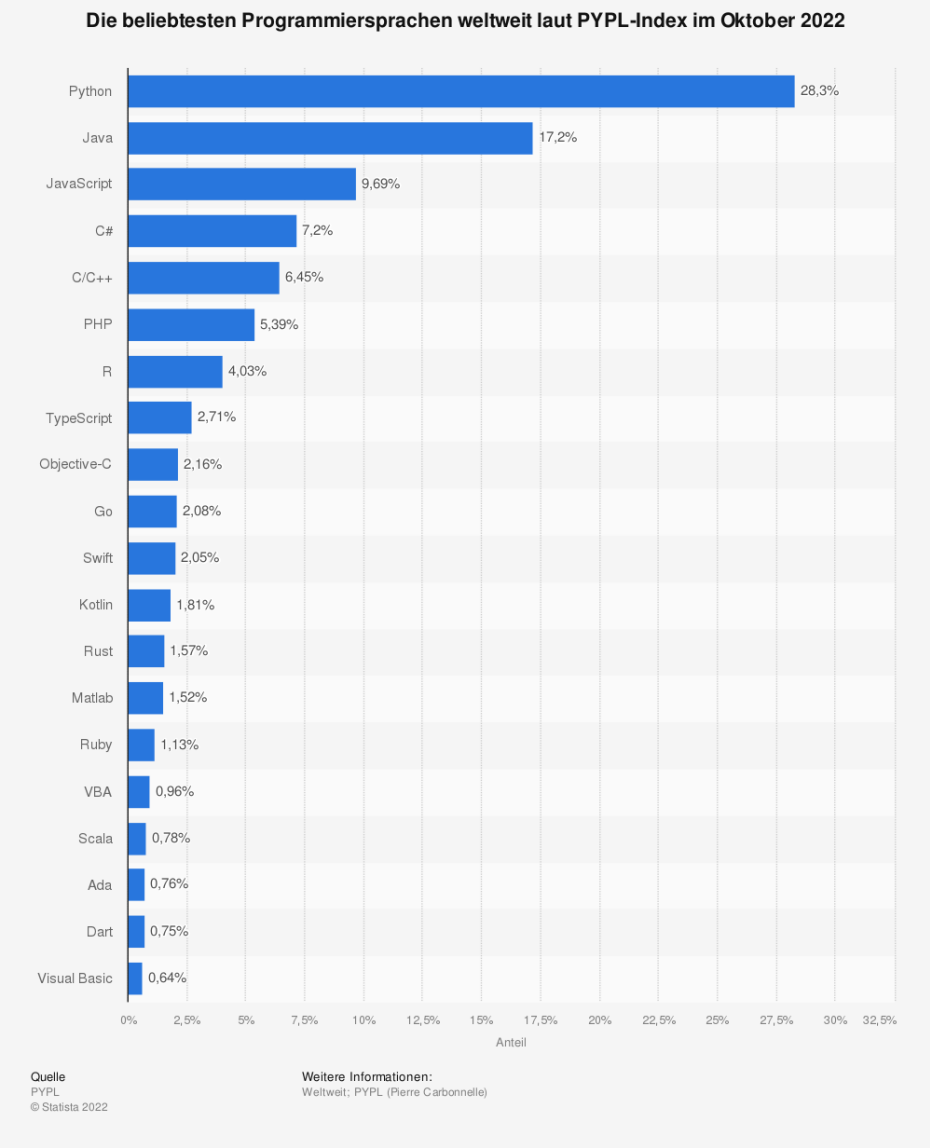
1993 – Abänderung der Zielstellung:
Entwicklung zur plattformunabhängigen,
zur Programmierung
von Internet-Applikationen geeigneten OOP

23. Mai 1995 – Veröffentlichung der Java-Technologie.

…

20. September 2022 – Veröffentlichung von Java 19

Verteilte Software - Java 17



Ranking-listen, z.B.
PYPL (Popularity of Programming Language Index)

- maßgeblich ist die Häufigkeit der Google Suchanfragen nach Tutorial
- weicht ab vom tatsächlichen Nutzen der Sprache

Analyseanbieter SlashData: Umfragen von mehr als 20000 Entwickler:innen aus 166 Länder :
JavaScript, Python, Java, C/C++, C#, PHP,…

niederländische Software-Unternehmen TIOBE: berücksichtigt die Häufigkeit der Anfragen
Quelle:
https://entwickler.de/programmierung/top-10-programmiersprachen-tiobe-pypl-devoloper-nation

Java – eine Erfolgsgeschichte

aber warum?

Java - Eigenschaften

plattformenunabhängig (Java – Bytecode)

objektorientierte Sprache (Klassen, Referenzen, virtuelle Methoden)

Sicherheit (Garbage Collection, Typüberprüfung zur Laufzeit)

Threads (Nebenläufigkeit)

Netzwerkfähigkeit (Sockets, RMI, JMS, Servlets)

modularer Aufbau (mehrere Dateien)

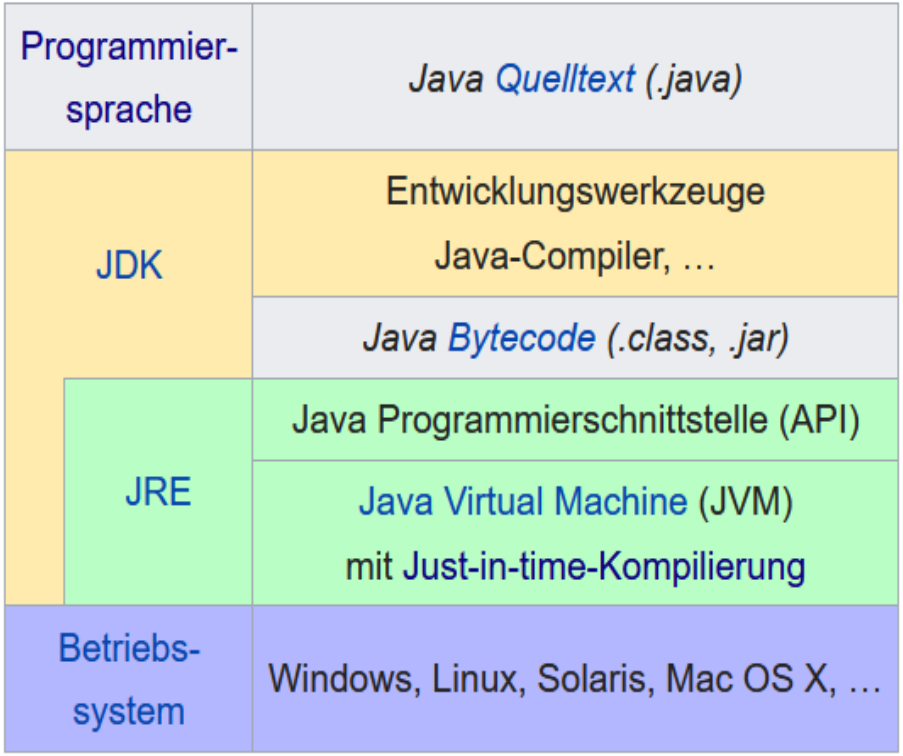
umfangreiche Klassenbibliotheken

Einbinden von Routinen anderer Programmiersprachen

Java-Technologie

- Programmiersprache
- Java Development Kit (JDK) – Entwicklungswerkzeuge …
  - Java-Compiler javac,
  - Java-Interpreter java,
  - JRE
  - Dokumentation javadoc
  - Debugger jdb
  - Bibliotheken
  - Archivierungstools
  - …
- Java-Laufzeitumgebung (JRE) – zum Ausführen der entwickelten Programme
  - Java Programmierschnittstelle – Bindung auf Quelltextebene

Aufbau der Java-Technologie



https://de.wikipedia.org/wiki/Java-Technologie

Erzeugen des Bytecodes: javac

javac name.java

Abarbeiten des Bytecodes: java, javaw (ohne Konsolenfenster)

java name

Entwicklungsumgebung - IDE (Integrated Development Environment)

- Eclipse
- NetBeans
- IntelliJ IDEA
- …