



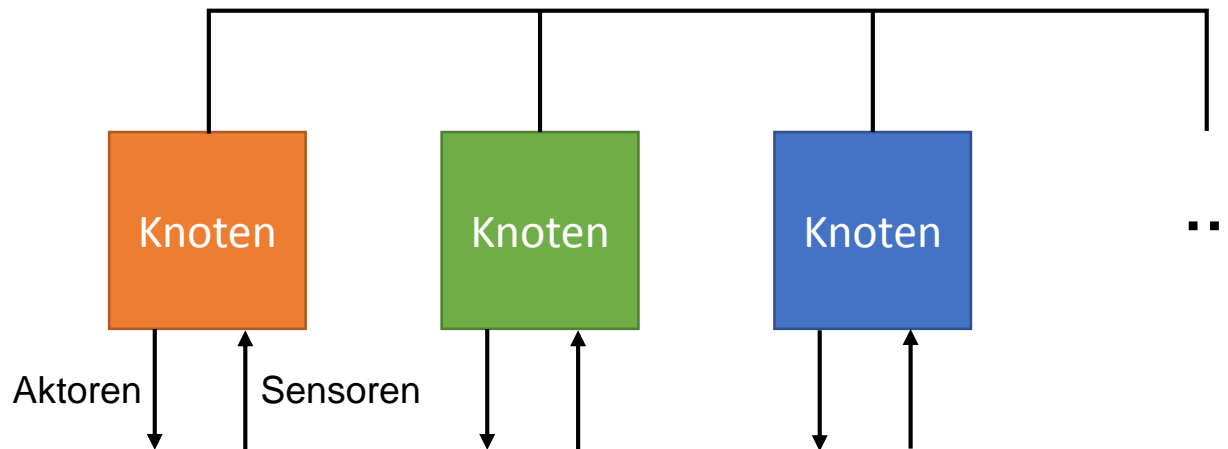
TECHNISCHE UNIVERSITÄT  
CHEMNITZ

Unterstützung mobiler Anwendungen  
mit verteilten aktiven Objekten

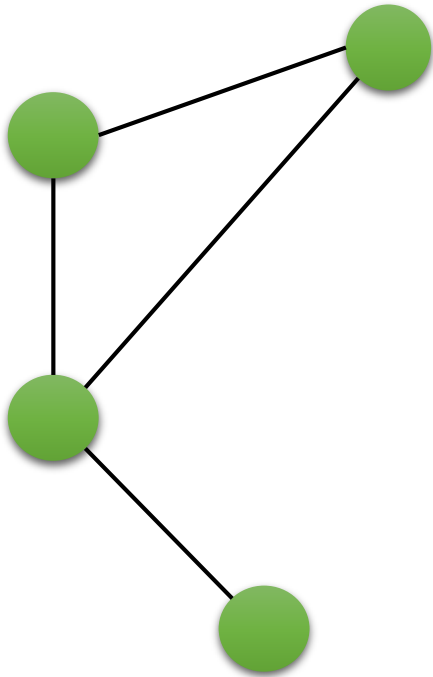
# Unterstützung mobiler Anwendungen mit verteilten aktiven Objekten

# Problem

- Echtzeit, Zuverlässigkeit und Diversität
- Explizite Kommunikation fehleranfällig und kompliziert
- Verteilte aktive Objekte als Lösung

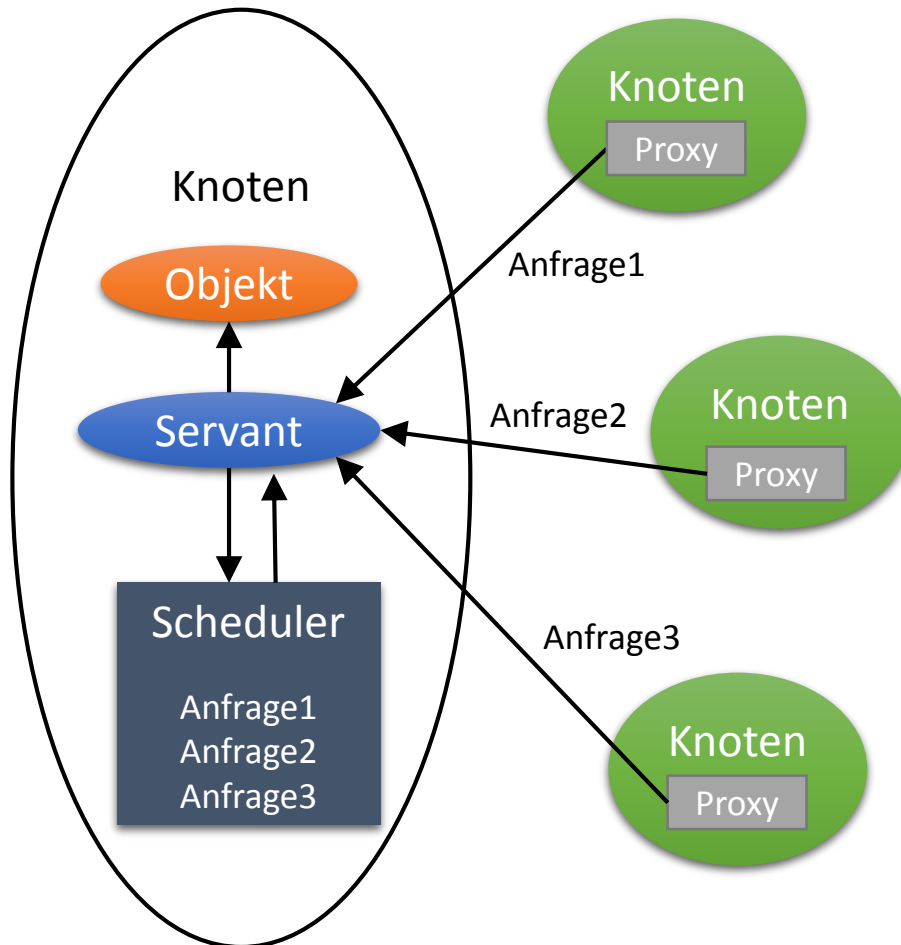


# Verteilte Systeme



- Mehrere miteinander verbundene Knoten
- Transparenz, Effizienz, Skalierbarkeit, Verfügbarkeit
- Abstraktion durch Middleware
- Automatisierte Versprechen von Eigenschaften

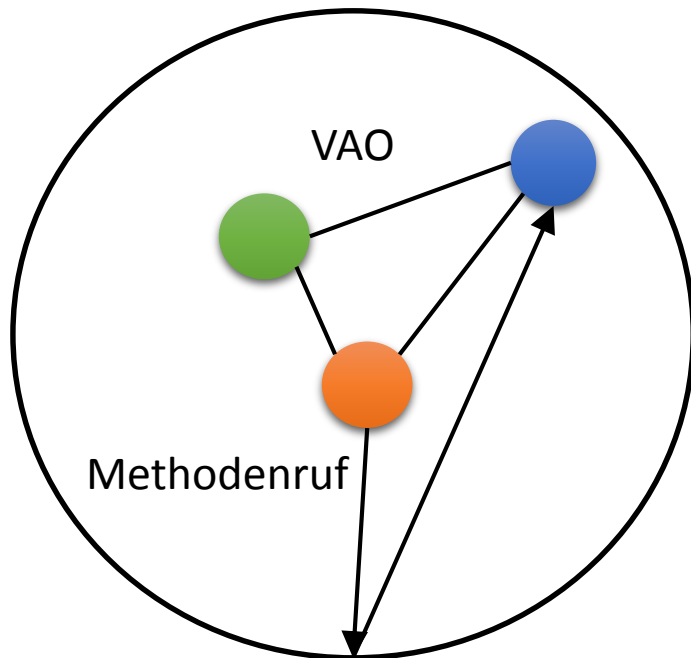
# Aktive Objekte



- Mehrere Knoten mit entfernten Objekten
- Trennung von Methodenruf und -ausführung
- Abstraktion von komplizierter Kommunikation
- Nebenläufigkeit und zeitlich abgestimmter Zugriff

Quelle: Greg R. Lavender and Douglas C. Schmidt. Active object - an object behavioral pattern for concurrent programming. 1996.

## Verteilte Aktive Objekte – Grundgedanke



- Verteiltes System als ein Objekt
- Gleichberechtigte Knoten
- Verteilter Kontrollfluss
- Steuerung des Kontrollflusses durch Constraints auf Basis des globalen Zustandes

## Verteilte Aktive Objekte – Ziel

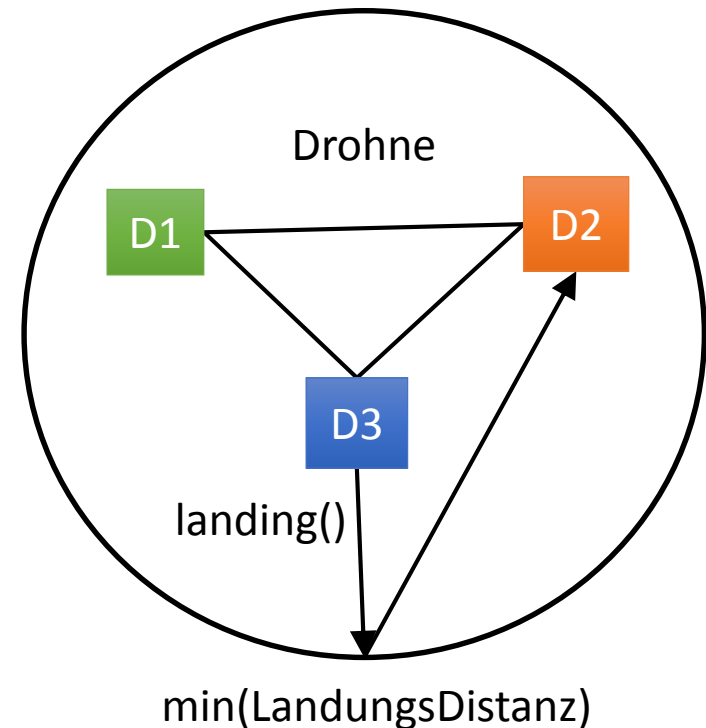
```

class Drohne {
    float LandungsDistanz;
    (int, int) LandePosition;

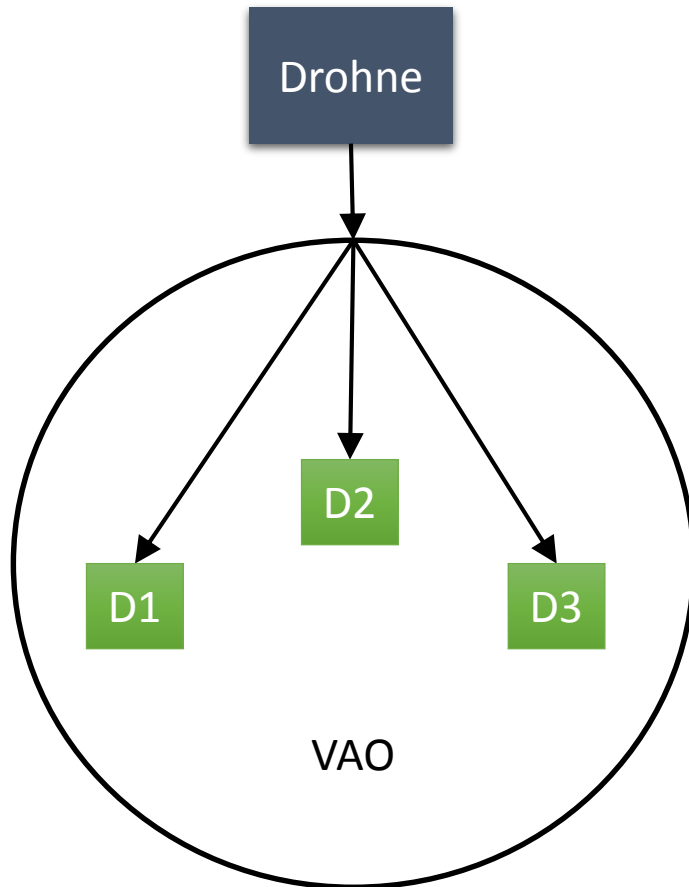
    @Require(min(LandungsDistanz))
    landing() { // lande }

    eventLoop() {
        // ...
        this.landing();
    }
}

```

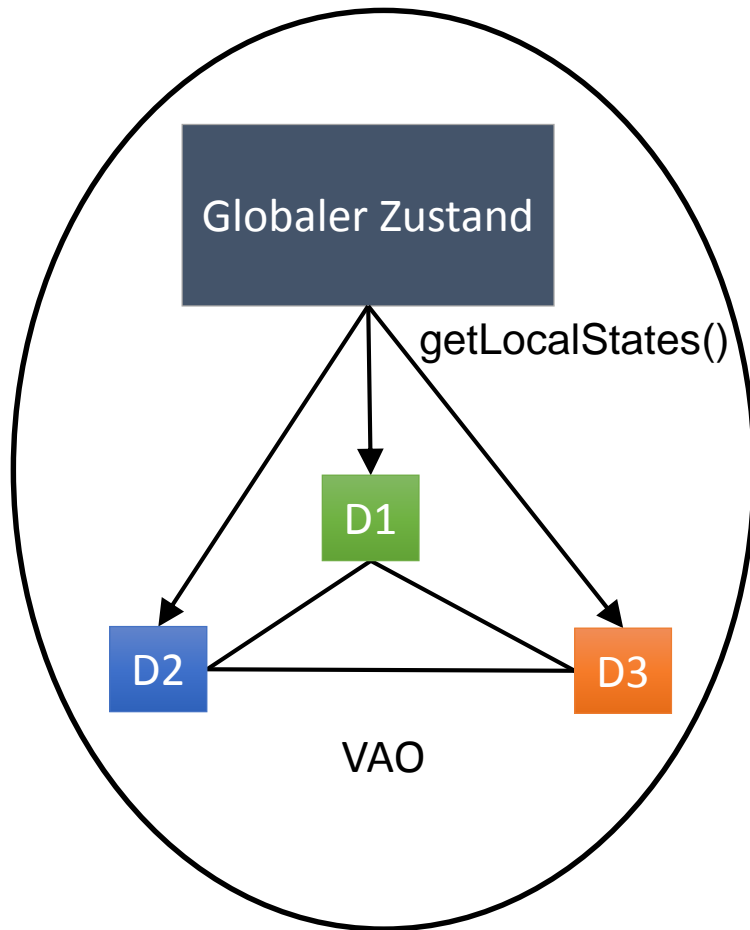


# Prototypische Umsetzung - Initialisierung



- Verteilung Klassendefinition
- Erreichbarkeit der Knoten untereinander durch RPC-Middleware
- Gegenseitige Zustandsänderungen durch Methodenrufe
- Änderung von Eigenschaften zur Laufzeit

# Prototypische Umsetzung - Globaler Zustand

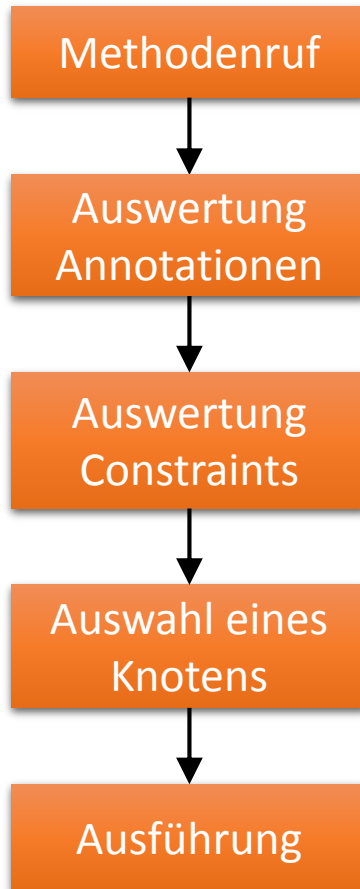


- Objekt, das periodisch lokale Zustände ausliest
- Speicherung Knoteninformationen in invertierter Liste

Attributname	Werte	Proxies
LandungsDistanz	13.4 2.1 15.6	Proxy2 Proxy1 Proxy3
LandePosition	(13, 14) (1, 5)	Proxy1, Proxy2 Proxy3



# Prototypische Umsetzung – Constraints



- Bindung an Methoden durch Annotationen
- Auswahl von Knoten nach seinen Eigenschaften
- Durch entfernte Methodenrufe verteilter Kontrollfluss

## Prototypische Umsetzung – Constraints

Attributname	Werte	Proxies
LandungsDistanz	13.4	Proxy2
	2.1	Proxy1
	15.6	Proxy3

[13.4, 2.1, 15.6]

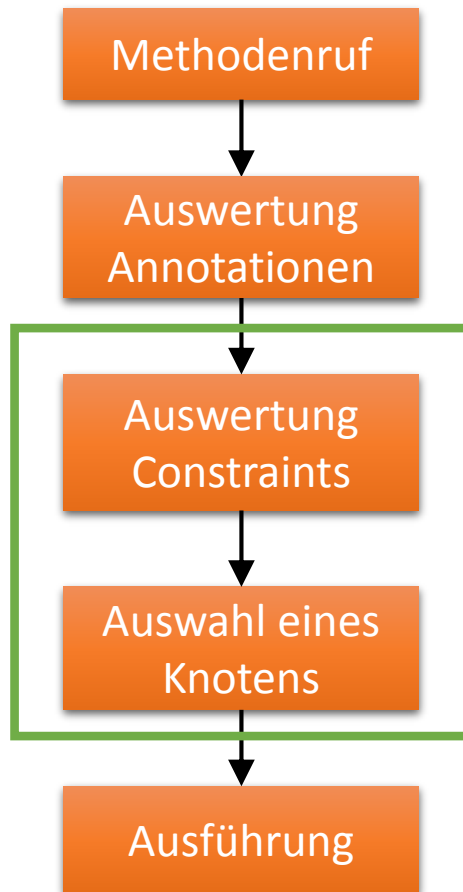
```
def minDist(self, ld : "LandungsDistanz"):
    return {"ld" : [min(ld)]}
```

```
@Require(minDist)
```

```
def landing(self):
```

```
    # Landung durchführen
```

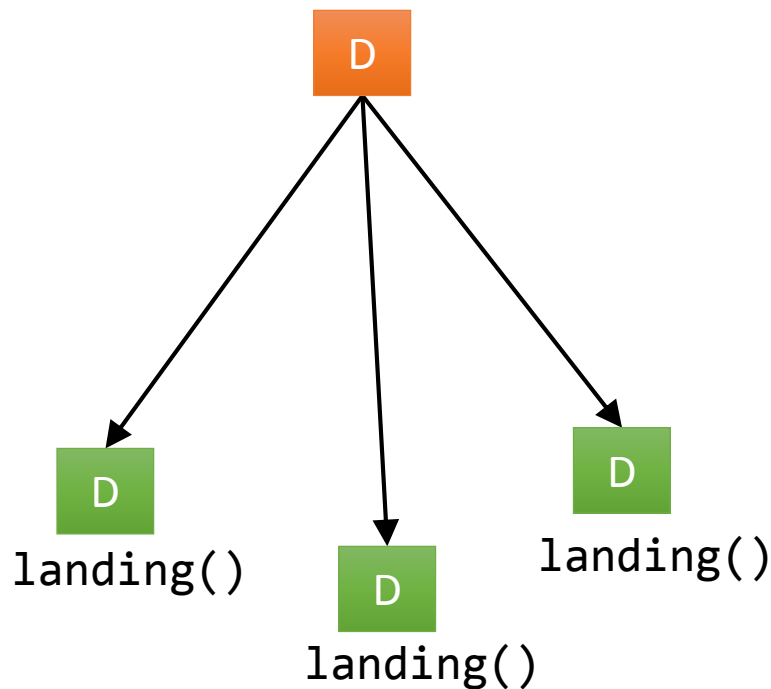
## Prototypische Umsetzung – Aktivierung



- Trennung von Ruf und Ausführung
- Einführung Zwischeninstanz
- Aktivierung durch Bindung von Parametern an Methodenruf

# Prototypische Umsetzung – Aktivierungsparameter

```
self.landing()#@count=3, sync, wait, once
```



- synchron/asynchron
- repeat/once
- force/wait
- count

## Prototypische Umsetzung – Aktivierung

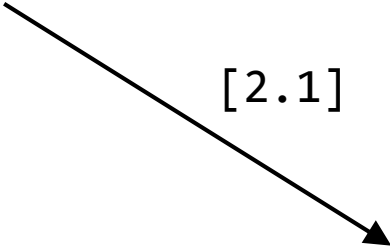
```
def minDist(self, ld : “LandungsDistanz“):
    return {“ld“ : [min(ld)]}
```

```
@Enable()
```

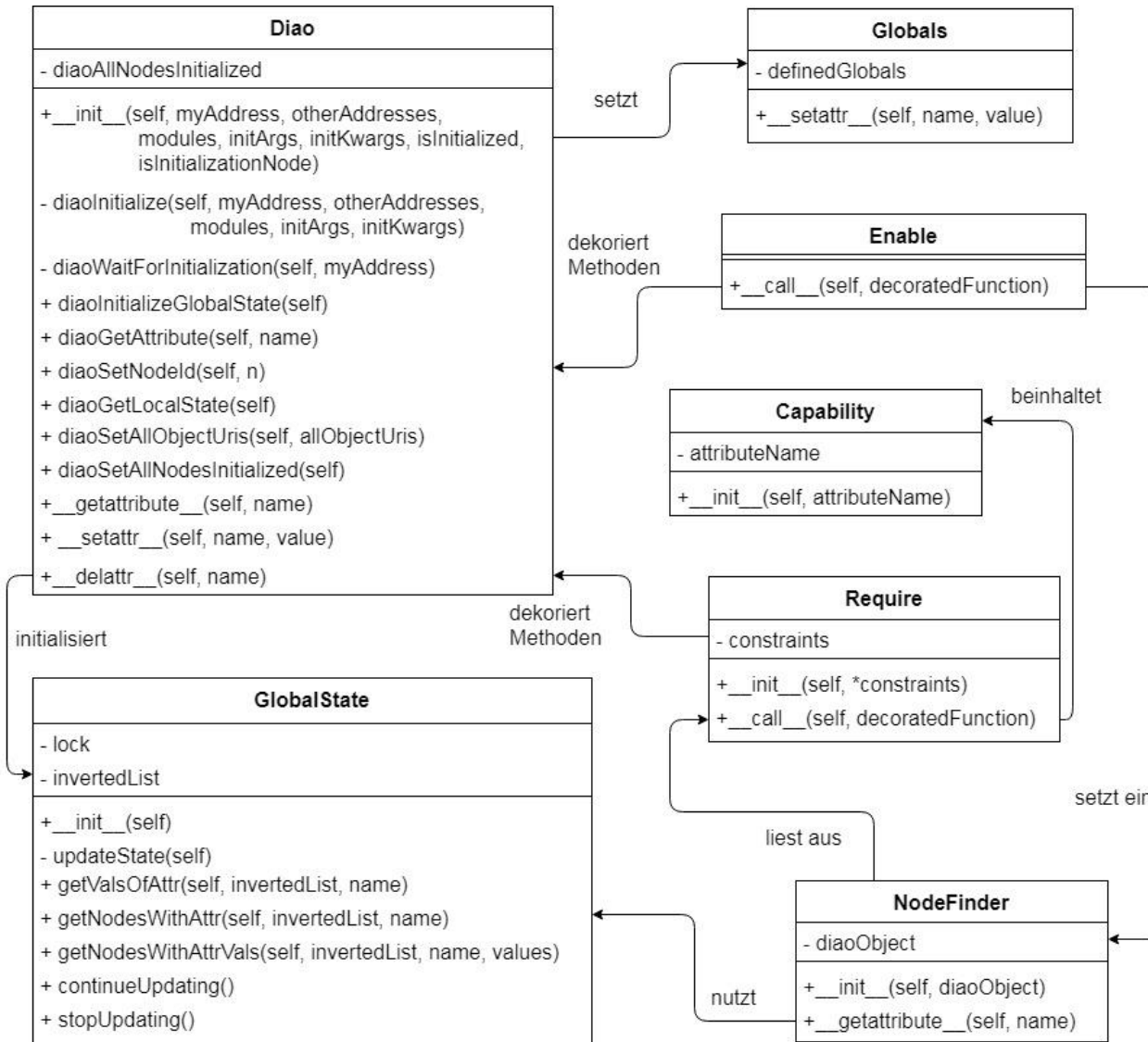
```
def eventLoop(self):
```

```
    self.landing()#@sync@LandungsDistanz=ld
```

[2.1]

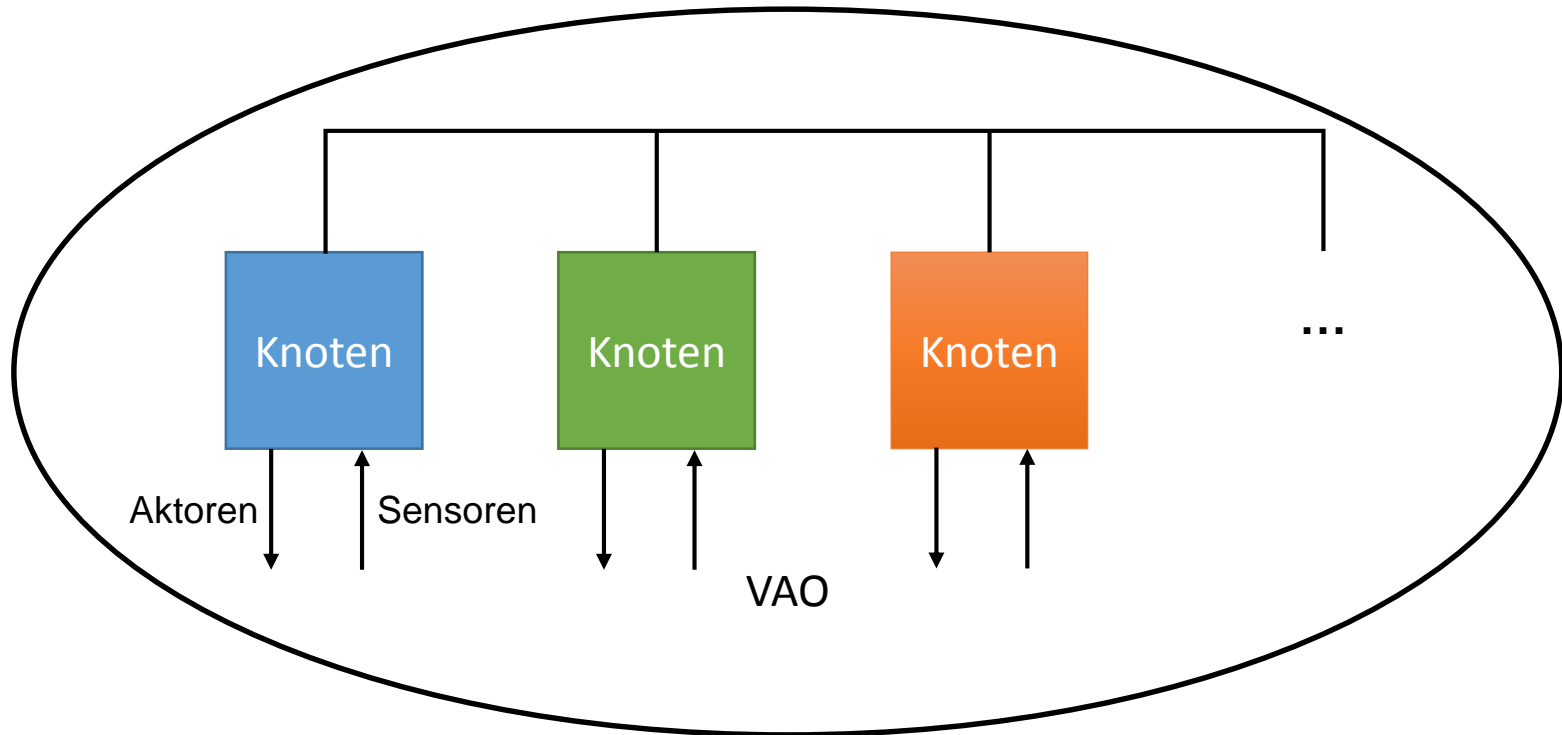


Attributname	Werte	Proxies
LandungsDistanz	13.4	Proxy2
	2.1	Proxy1
	15.6	Proxy3



# Zusammenfassung

- Verringerung Kompliziertheit der Knotenkoordination
- Abstraktion von Verteiltheit



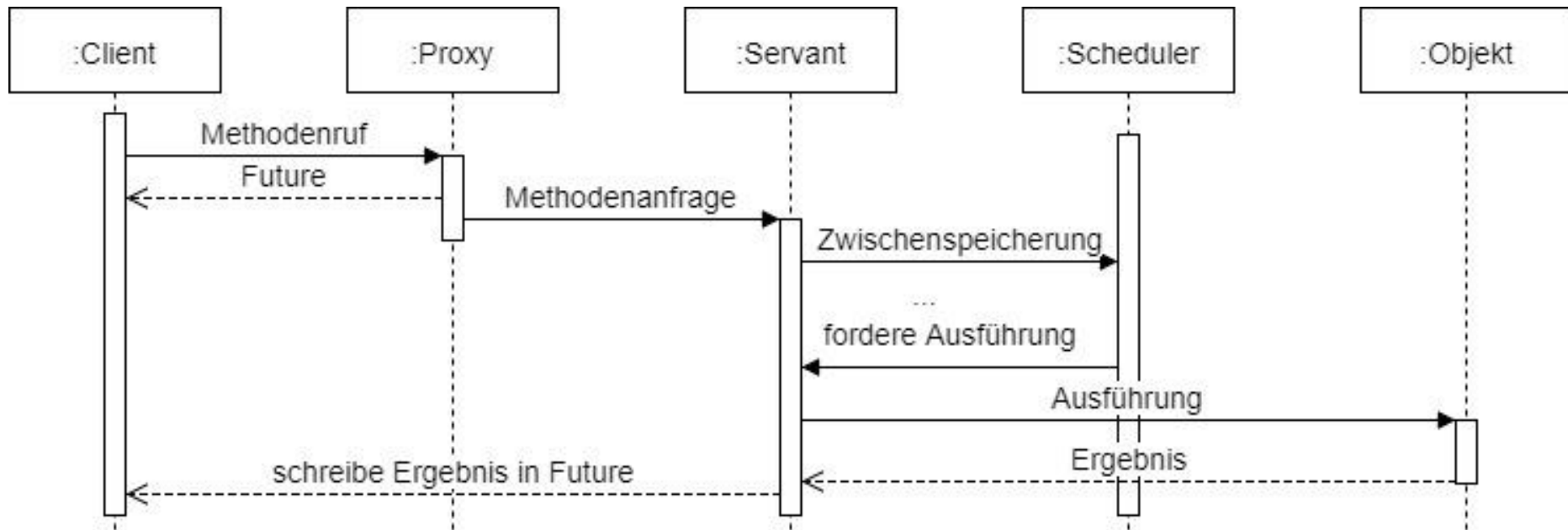
## Quellen

- Matthias Werner and Peter Tröger. Verteilte Betriebssysteme. Kapitel 2. Charakterisierung verteilter Systeme. [http://osg.informatik.tu-chemnitz.de/lehre/dos/dos-02-Charakterisation-handout\\_de.pdf](http://osg.informatik.tu-chemnitz.de/lehre/dos/dos-02-Charakterisation-handout_de.pdf). Last Accessed: 29.10.2017.
- Matthias Werner and Peter Tröger. Verteilte Betriebssysteme. Kapitel 3. Architekturen. [http://osg.informatik.tu-chemnitz.de/lehre/dos/dos-03-Architekturen-handout\\_de.pdf](http://osg.informatik.tu-chemnitz.de/lehre/dos/dos-03-Architekturen-handout_de.pdf). Last Accessed: 29.10.2017.
- Daniel Graff, Jan Richling, Tammo M. Stupp, and Matthias Werner. 8th iee international conference and workshops on engineering of autonomic and autonomous systems. In Distributed Active ObjectsA Systemic Approach to Distributed Mobile Applications, 2011.
- Greg R. Lavender and Douglas C. Schmidt. Active object - an object behavioral pattern for concurrent programming. 1996.

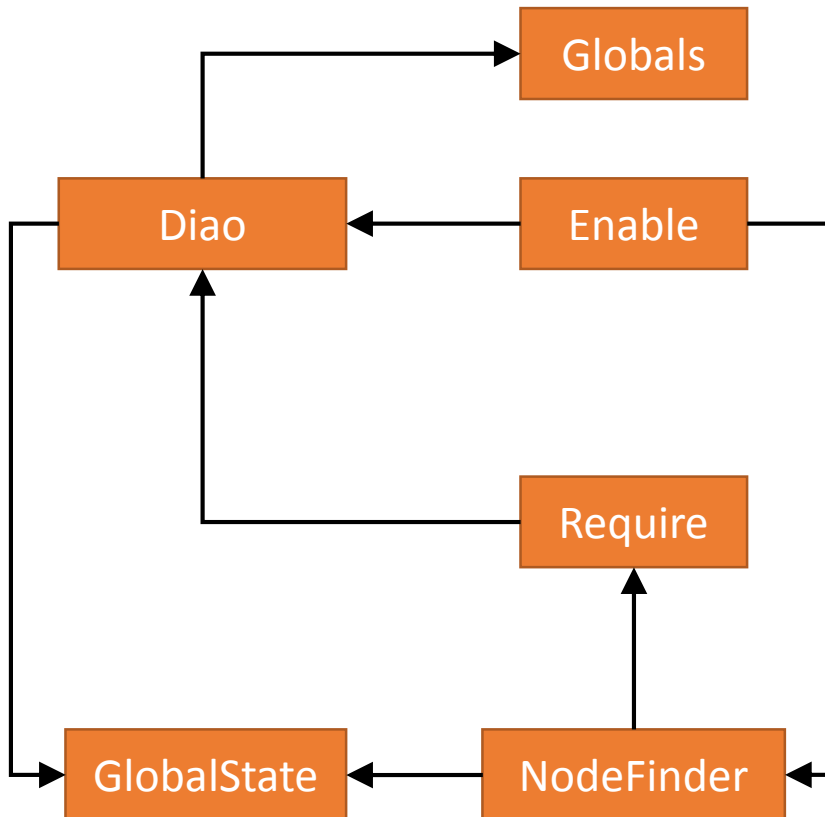


# Aktive Objekte

- Asynchrone Methodenrufe
- Trennung von Ruf und Ausführung einer Methode



# Überblick



- Python 3.6
- Python Remote Objects (PyRO)
- Dekoratoren
- Parameter-Annotationen