Universität Erlangen-Nürnberg

Lehrstuhl für Rechnernetze und Kommunikationssysteme:

Echtzeitaspekte bei der Aufgabenverteilung in selbst-organisierenden autonomen Systemen

ROSES

Gerhard Fuchs, Falko Dressler

gerhard.fuchs@informatik.uni-erlangen.de

Friedrich-Alexander-Universität Erlangen-Nürnberg
TECHNISCHE FAKULTÄT

PEARL 2005

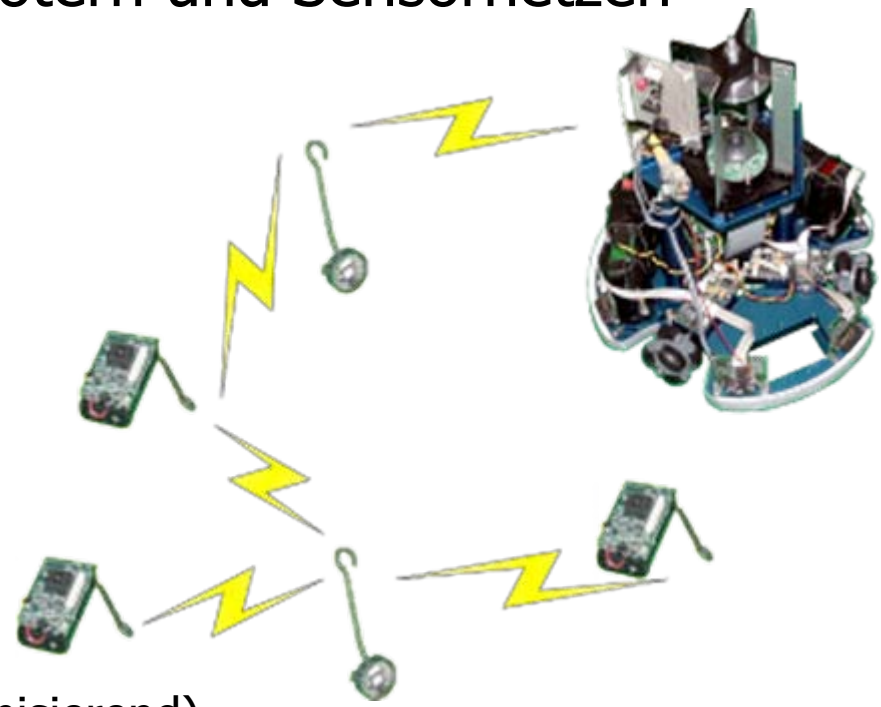
01.12.2005

Inhalt:

- ROSES-Projekt
- Szenario
- Prinzip der Zeitmessung
- Umsetzung
  - Allgemein
  - Experiment
  - Simulation
- Beispiele (MURDOCH / OAA)
- Zusammenfassung / Ausblick

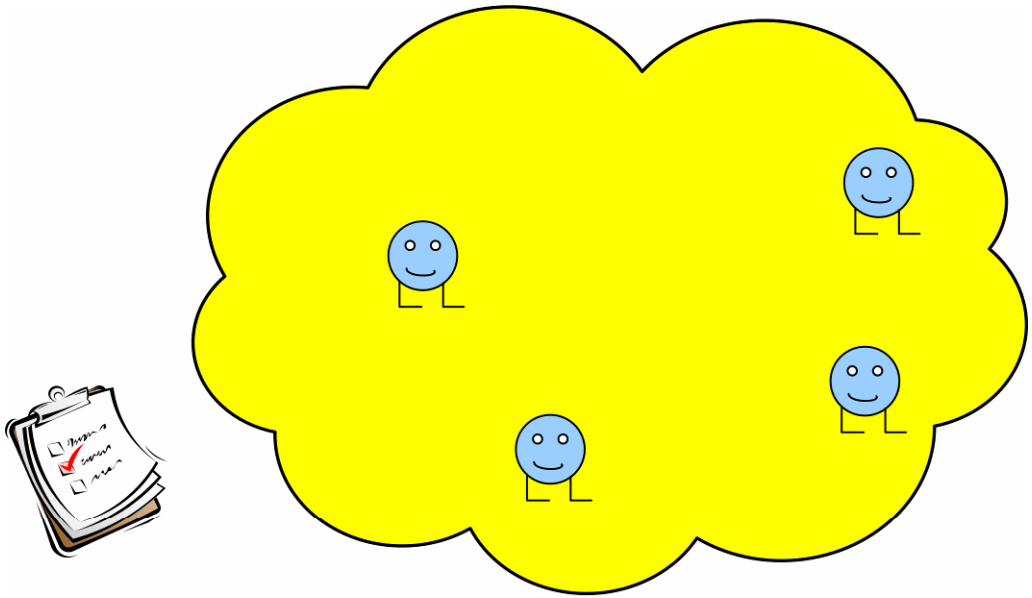
ROSES-Projekt:

- Kombination aus mobilen Robotern und Sensornetzen
  - ➔ **Mobile Sensornetze**



- Forschungsziele:
  - Kommunikation (QoS, Sicherheit …)
  - Sensornetzwerk-unterstützte Navigation / Lokalisierung
  - Koordination (autonom, selbst-organisierend)
  - **Echtzeitaspekte** (Einhalten von Zeitschranken) insbesondere bei der Aufgabenverteilung

Szenario – Ausgangslage:

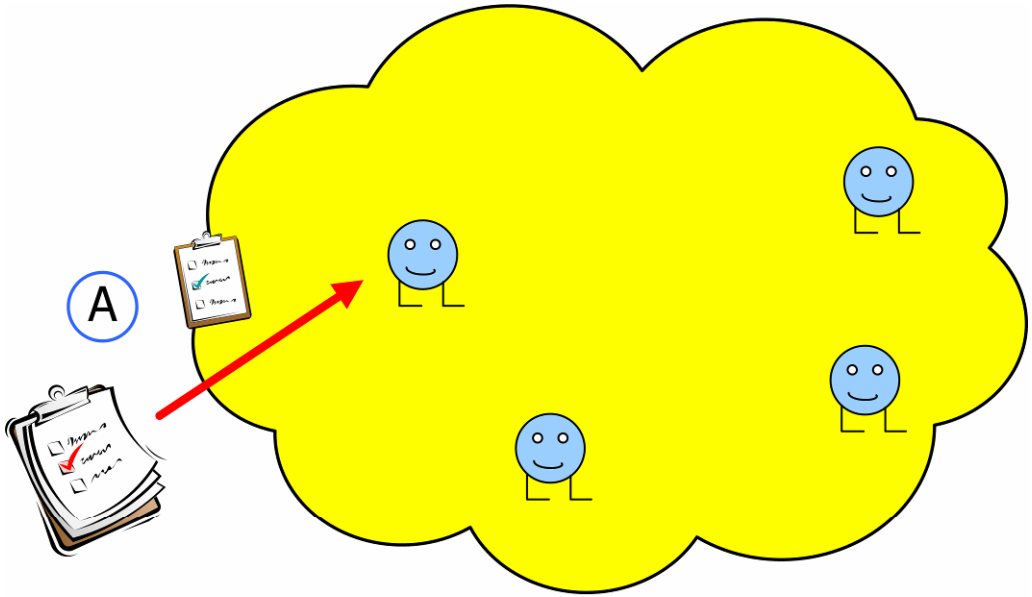


Roboter

Aufgabengenerator (AG)

- Selbst-organisierendes autonomes System
- Aufgabengenerator
- Roboter
  - Sind autonom
  - Erfüllen max. eine Aufgabe
  - Können Aufgaben ver-schieden gut lösen
  - Keine Neuvergabe
- Aufgaben
  - Verschiedene Typen
  - Sind für genau einen Roboter (keine Teams)

Szenario:

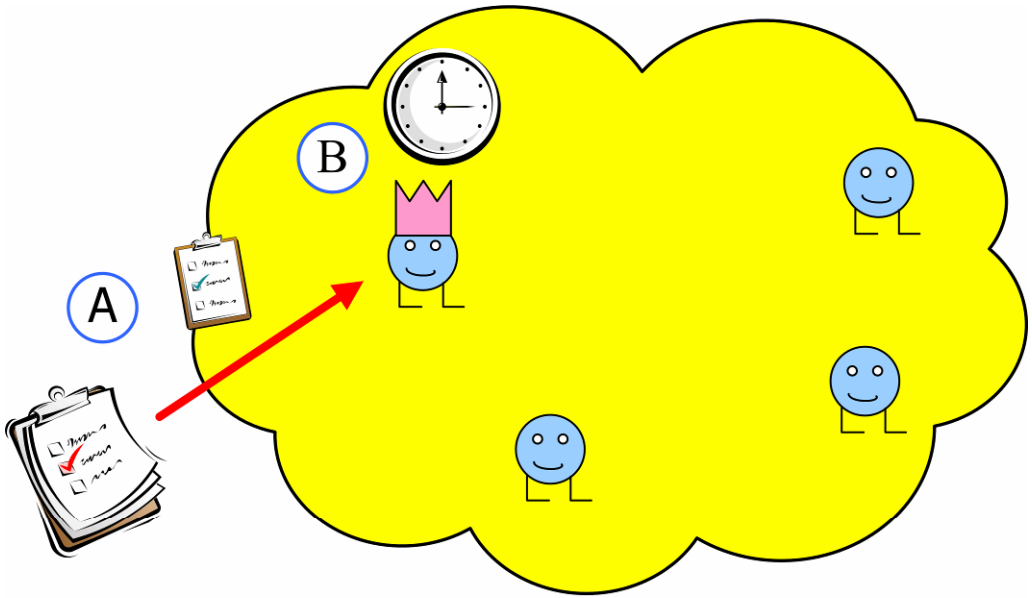


- Aufgabengenerierung
  - AG generiert Aufgabe (A)
  - AG sendet Aufgabe an bel. Roboter im System

Roboter

Aufgabengenerator (AG)

Szenario:



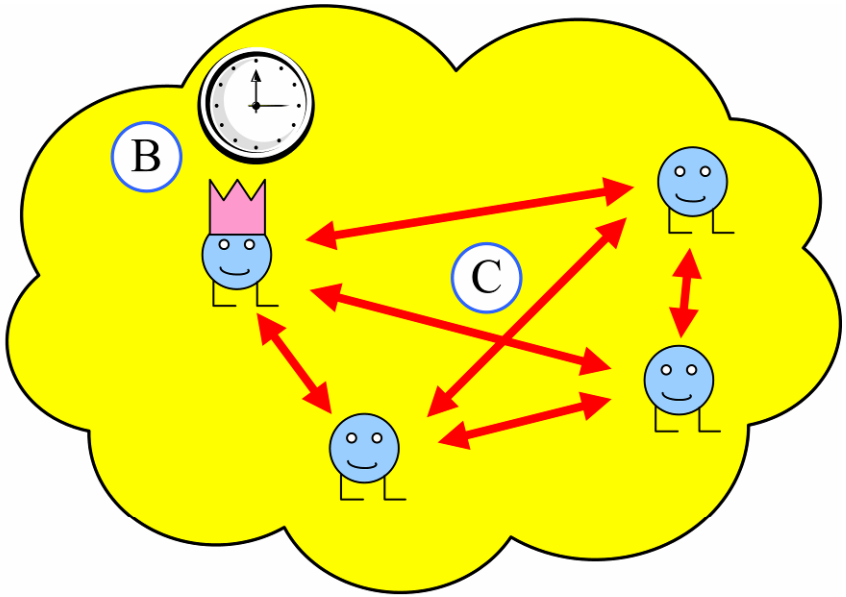
- Aufgabengenerierung
  - AG generiert Aufgabe (A)
  - AG sendet Aufgabe an bel. Roboter im System
- Roboter → Koordinator
  - Start der Zeitmessung (B)
  - Aufgabenverteilung


Roboter


Koordinator


Aufgabengenerator (AG)

Szenario:



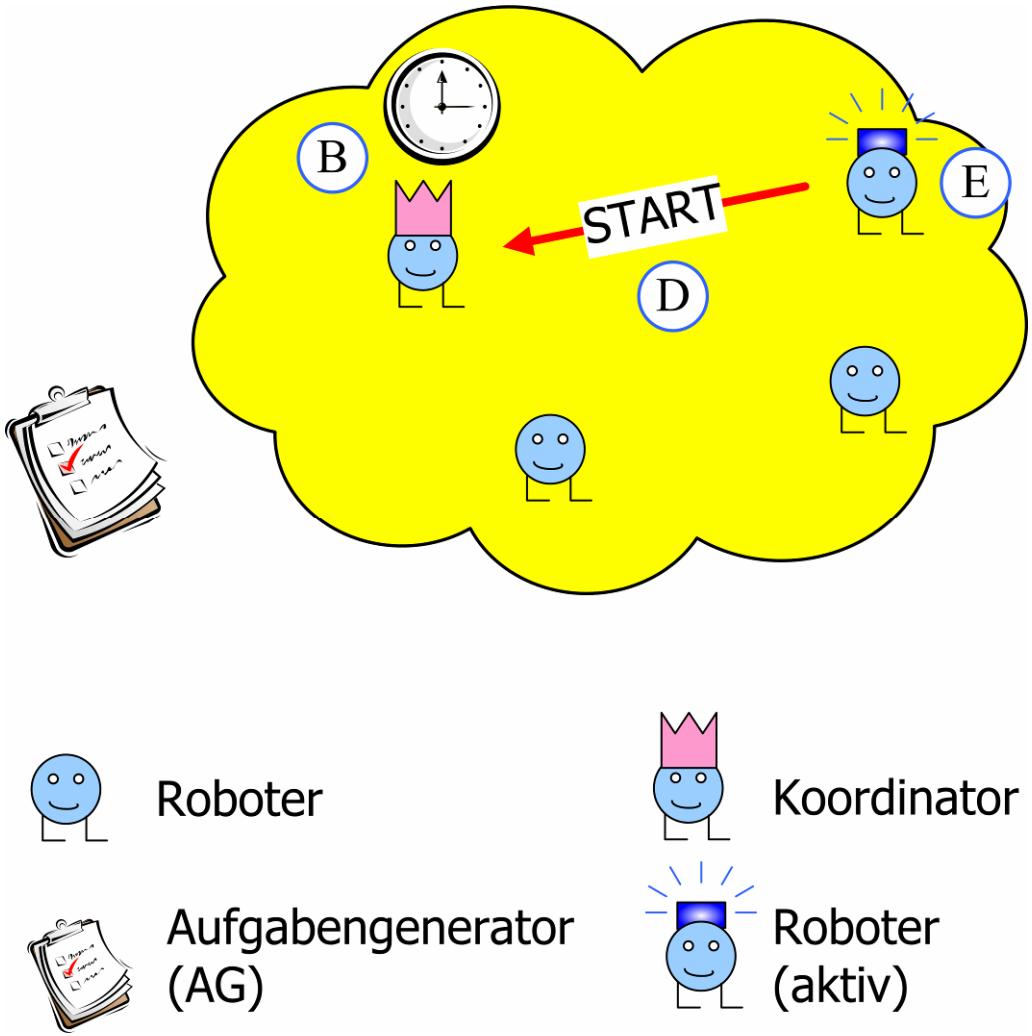
- Aufgabengenerierung
  - AG generiert Aufgabe (A)
  - AG sendet Aufgabe an bel. Roboter im System
- Roboter → Koordinator
  - Start der Zeitmessung (B)
  - Aufgabenverteilung (C)

Roboter

Koordinator

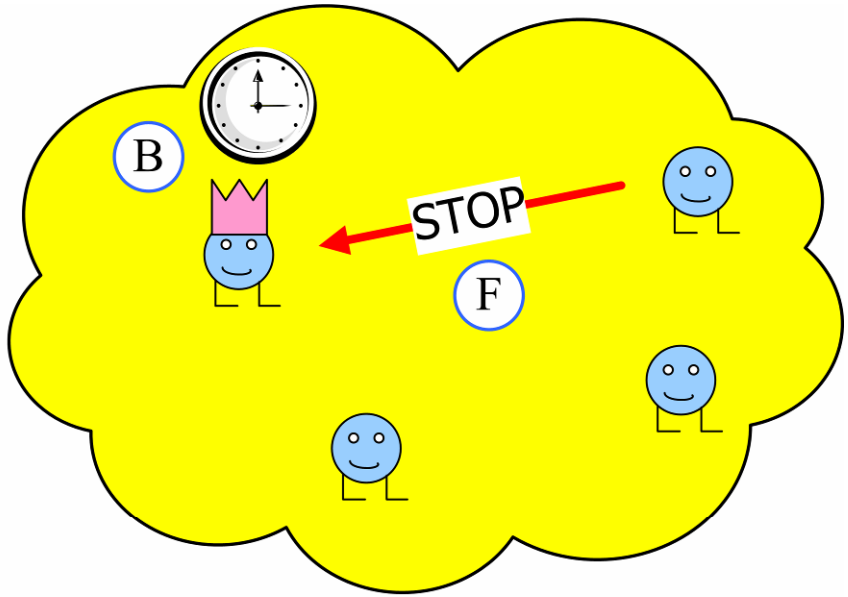
Aufgabengenerator (AG)

Szenario:



- Aufgabengenerierung
  - AG generiert Aufgabe (A)
  - AG sendet Aufgabe an bel. Roboter im System
- Roboter → Koordinator
  - Start der Zeitmessung (B)
  - Aufgabenverteilung (C)
- Ausführung
  - Start → Anzeigen (D)
  - Aktive Phase (E)

Szenario:



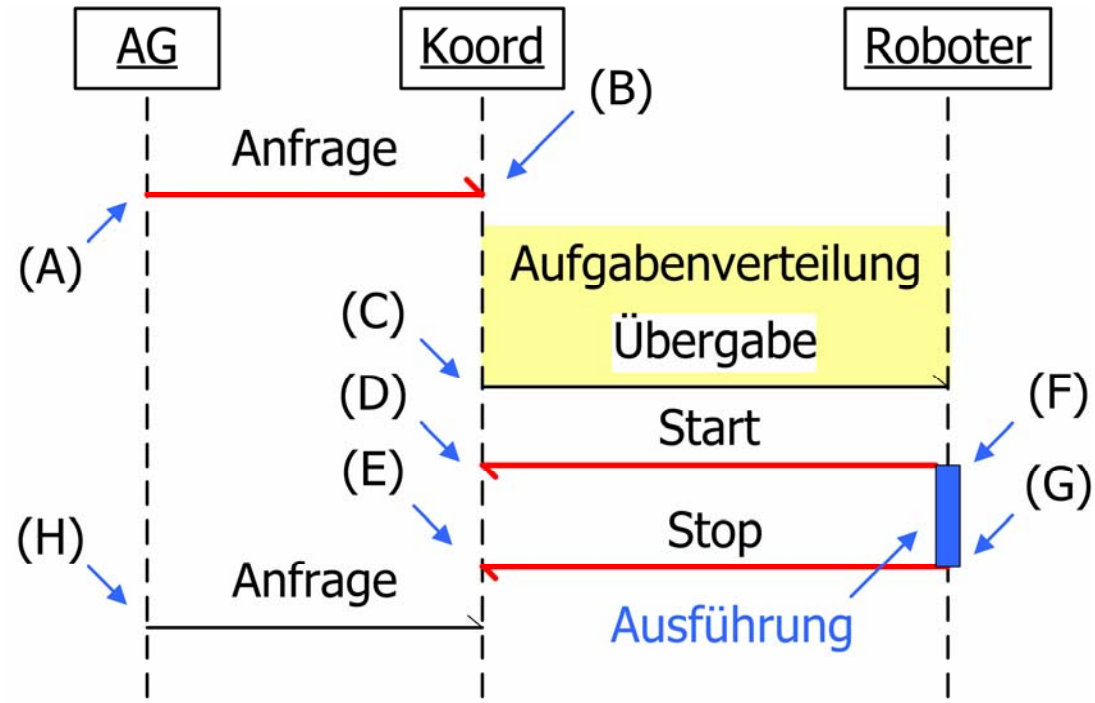
Roboter

Koordinator

Aufgabengenerator (AG)

- Aufgabengenerierung
  - AG generiert Aufgabe (A)
  - AG sendet Aufgabe an bel. Roboter im System
- Roboter → Koordinator
  - Start der Zeitmessung (B)
  - Aufgabenverteilung (C)
- Ausführung
  - Start → Anzeigen (D)
  - Aktive Phase (E)
- Abschluss (F)
  - Ende → Anzeigen

Prinzip der Zeitmessung:

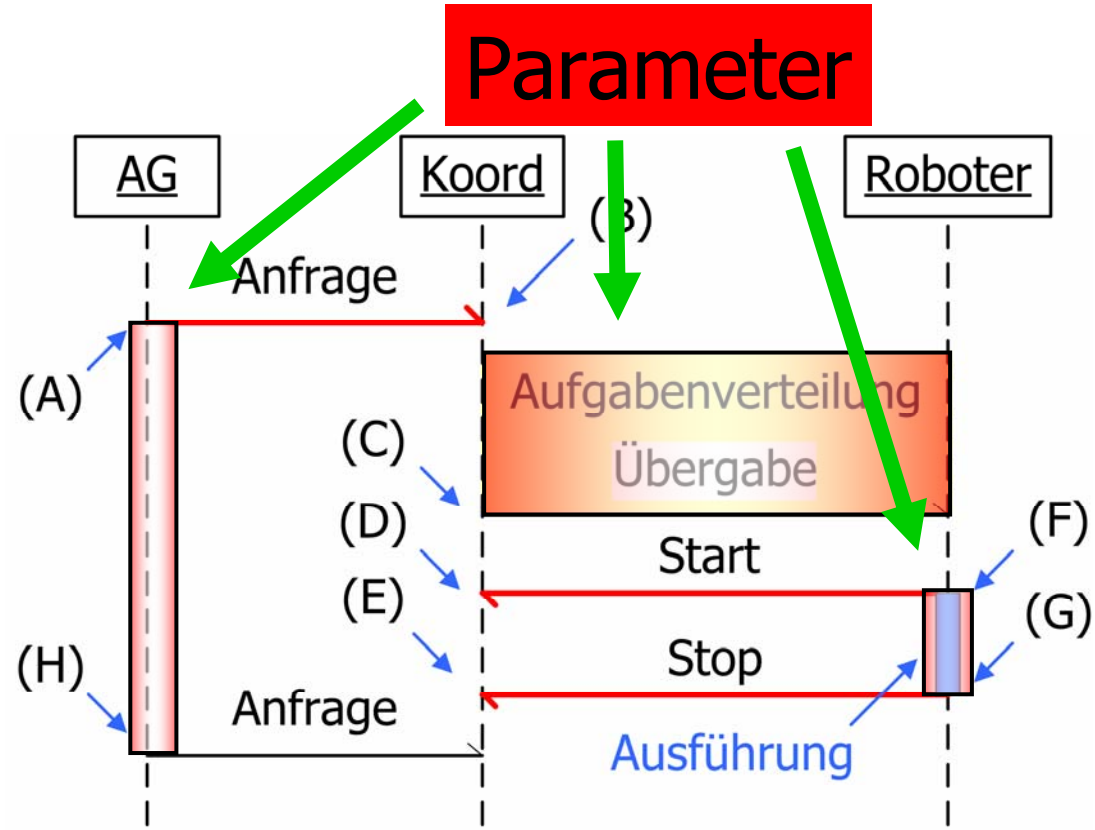


AG: Aufgabengenerator
Koord: Koordinator

- Für das Messen sind das Anfrage-, das Start- und das Stop-Signal relevant
- Aufgabenverteilung ist austauschbar
- Übergabe-Signal nicht unbedingt nötig

➔ **FLEXIBEL**

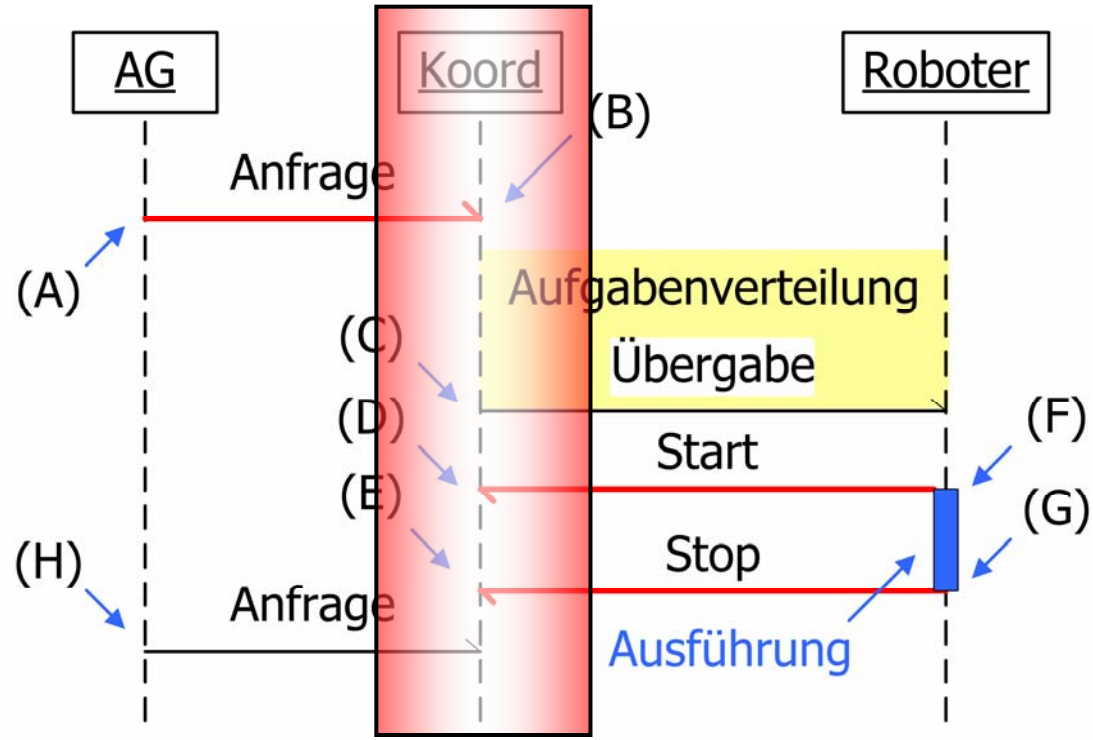
Prinzip der Zeitmessung :



AG: Aufgabengenerator
Koord: Koordinator

- Parameter
  - Hängen von Experiment und Mechanismus ab

Prinzip der Zeitmessung :



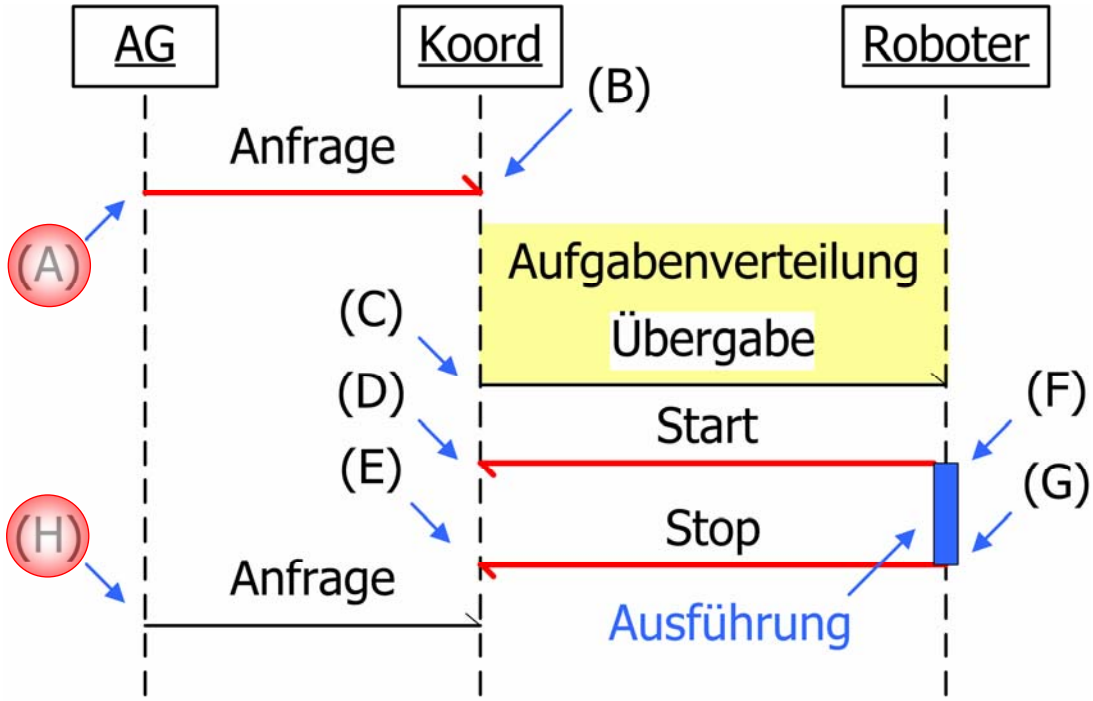
AG: Aufgabengenerator
Koord: Koordinator

Resultate

- Parameter
  - Hängen von Experiment und Mechanismus ab
- Resultate
  - ausschließlich beim Koordinator
  - Verwendung der lokalen Uhr möglich
  - Einfache Berechung von Differenzen

➔ **Keine Synchro-nisation der Knoten nötig**

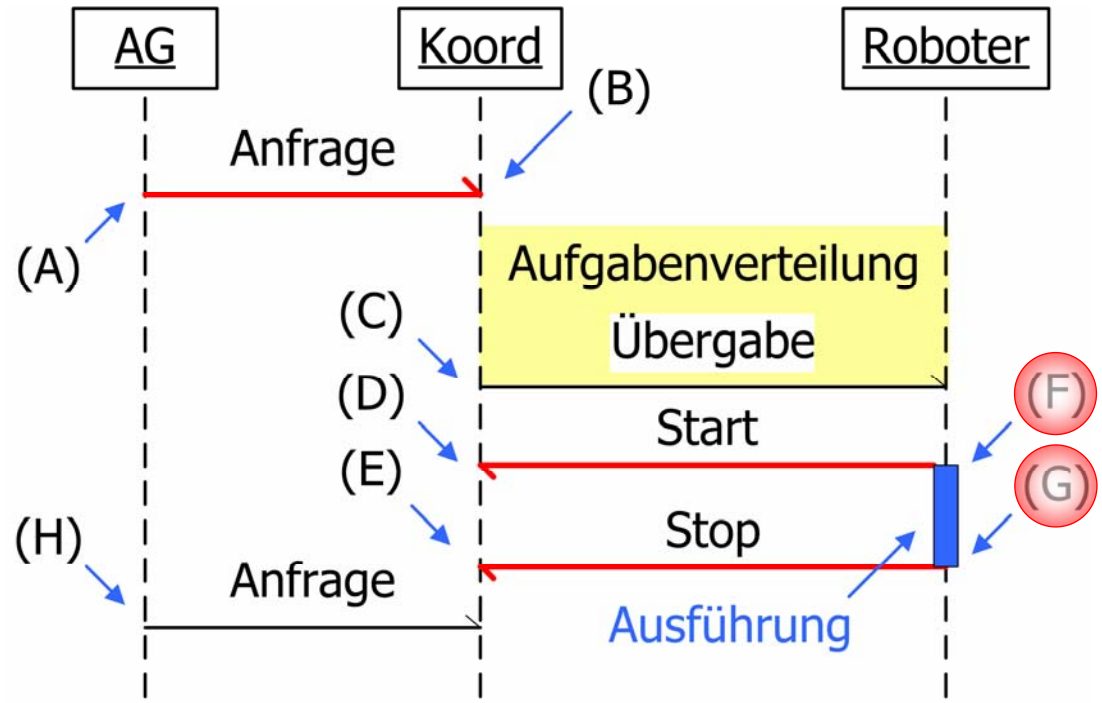
Prinzip der Zeitmessung – Parameter:



AG: Aufgabengenerator
Koord: Koordinator

- Anfrageabstand
  - T_{ANF}
  - Zeit zw. (A) und (H)

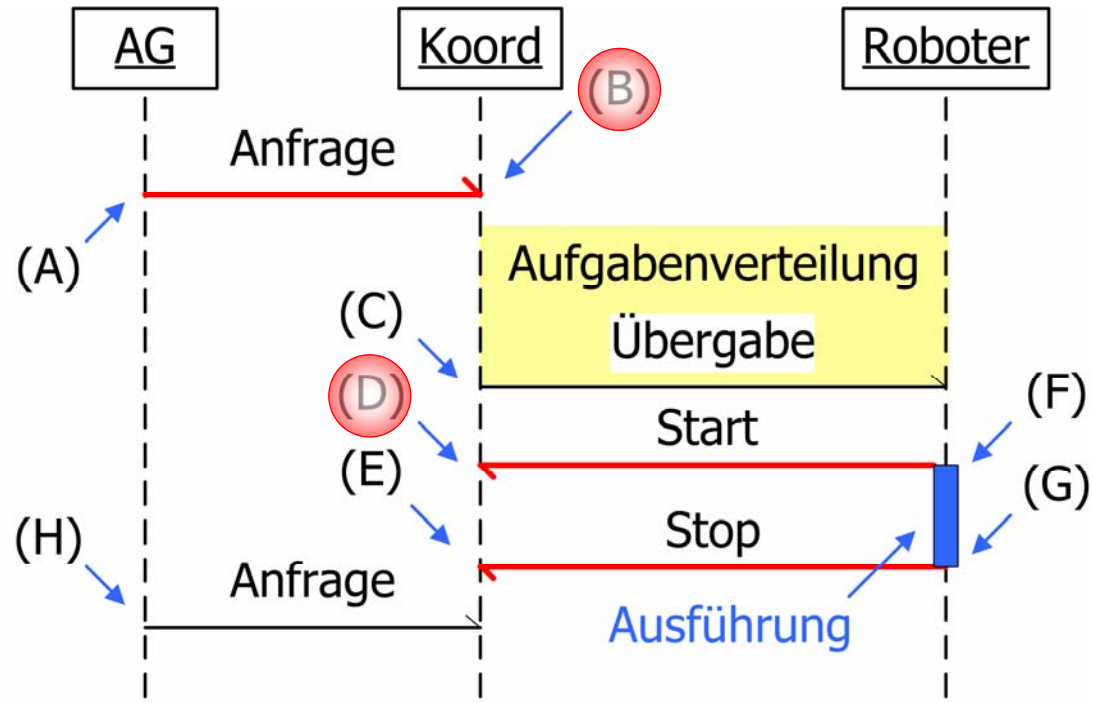
Prinzip der Zeitmessung – Parameter:



AG: Aufgabengenerator
Koord: Koordinator

- Anfrageabstand
  - T_{ANF}
  - Zeit zw. (A) und (H)
- Ausführungszeit
  - T_{EX}
  - Zeit zw. (F) und (G)

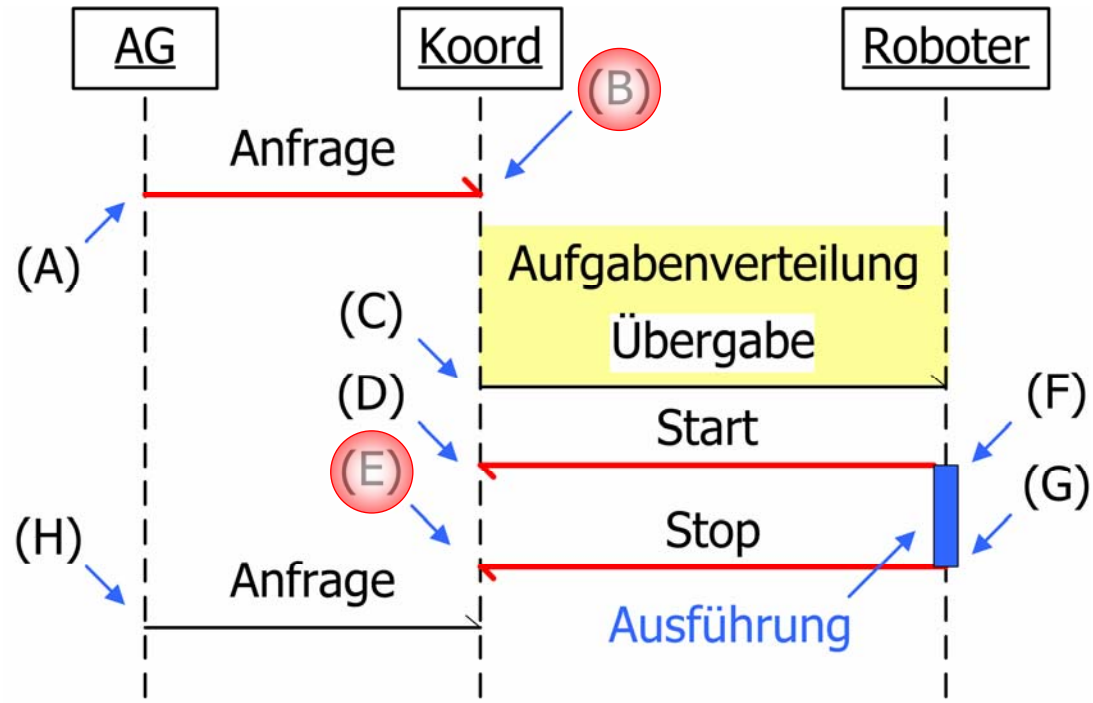
Prinzip der Zeitmessung – Resultate:



AG: Aufgabengenerator
Koord: Koordinator

- Anfrageabstand
  - T_{ANF}
  - Zeit zw. (A) und (H)
- Ausführungszeit
  - T_{EX}
  - Zeit zw. (F) und (G)
- Aufgabenverteilzeit
  - T_{AV}
  - Zeit zw. (B) und (D)

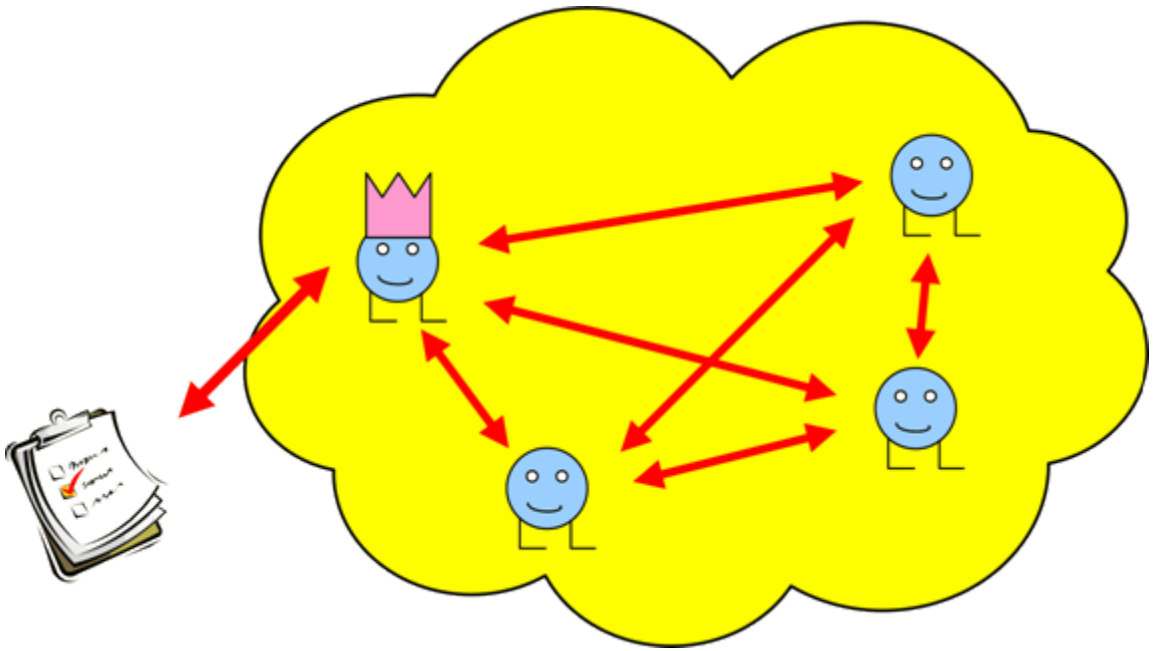
Prinzip der Zeitmessung – Resultate:



AG: Aufgabengenerator
Koord: Koordinator

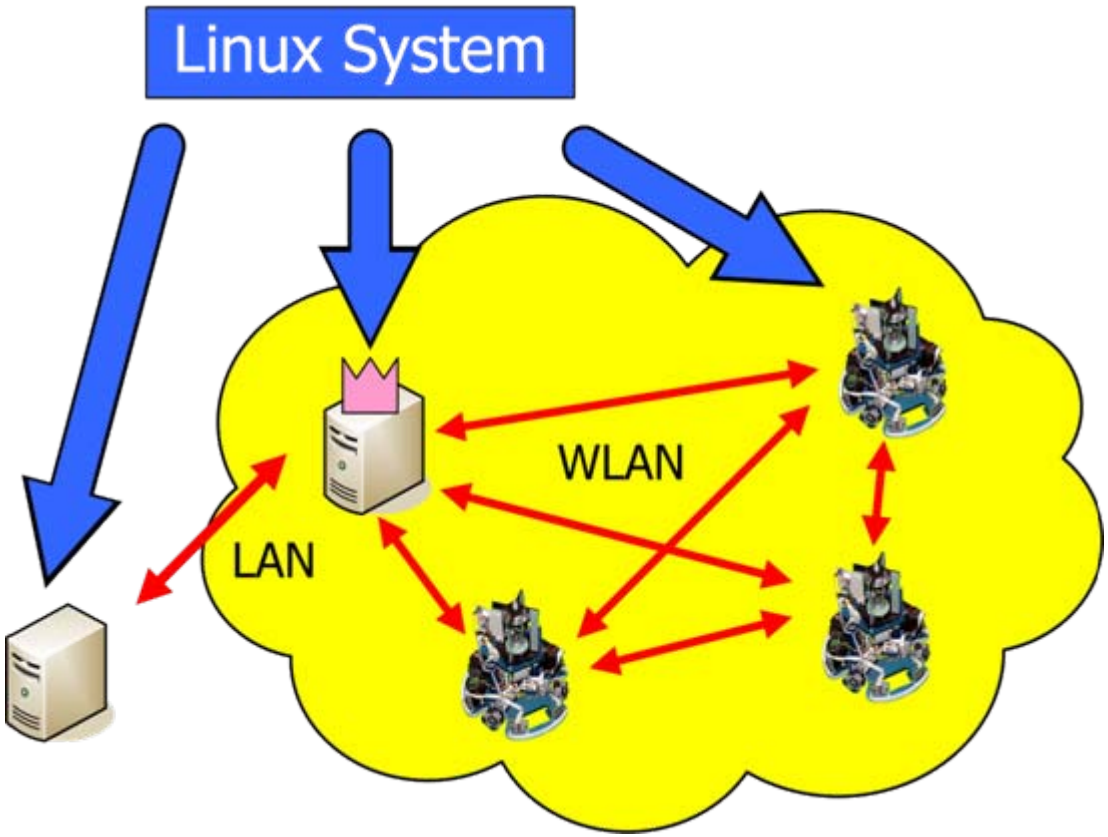
- Anfrageabstand
  - T_{ANF}
  - Zeit zw. (A) und (H)
- Ausführungszeit
  - T_{EX}
  - Zeit zw. (F) und (G)
- Aufgabenverteilzeit
  - T_{AV}
  - Zeit zw. (B) und (D)
- Verweildauer
  - T_V
  - Zeit zw. (B) und (E)

Umsetzung:



- Autonomes System
  - Aufgabengenerator
  - Koordinator
  - Roboter
- Aufgabenverteilung
  - Zwischen Koordinator und Roboter
- Mechanismus
  - Realisiert bei Koordinator und Robotern

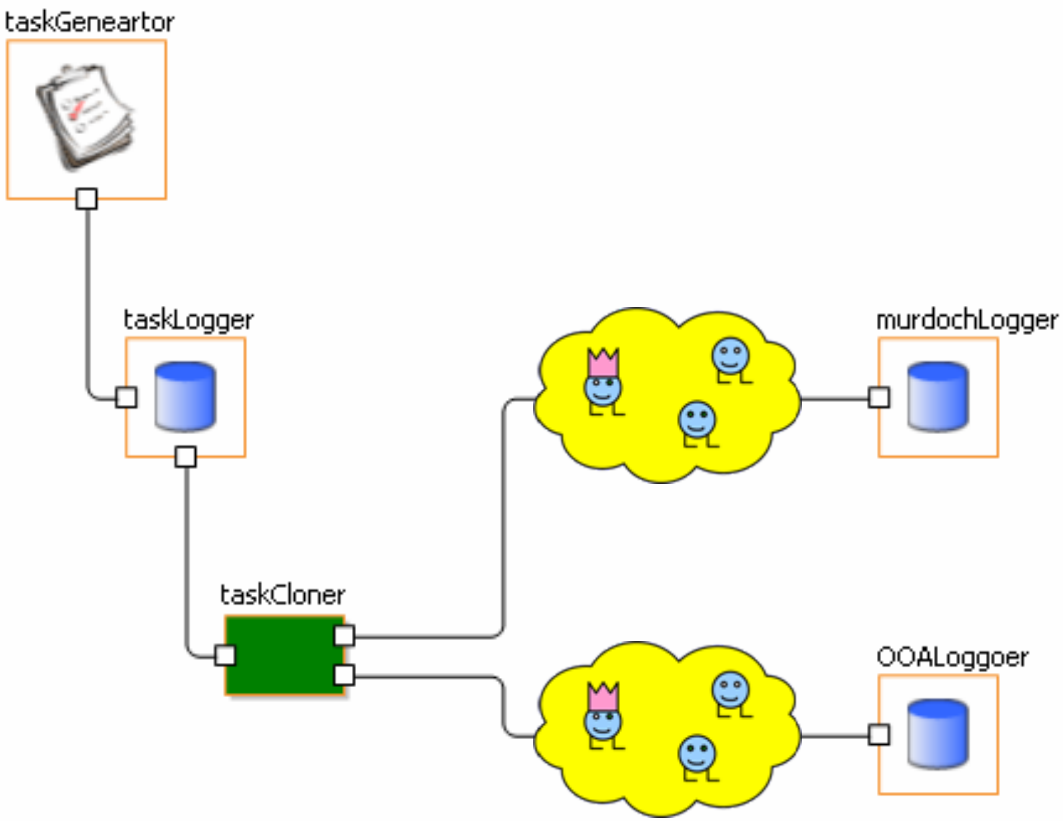
Umsetzung – Experiment:



- Aufgabengenerator
  - Handelsüblicher PC
- Koordinator
  - Handelsüblicher PC
- Roboter
  - Robertino – Plattform
  - „Linux-PC auf Rädern“
  - PC-104
  - Pentium III 500 MHz
  - 128 MB RAM
  - 20 GB HD

➔ **Programmierung in C++**

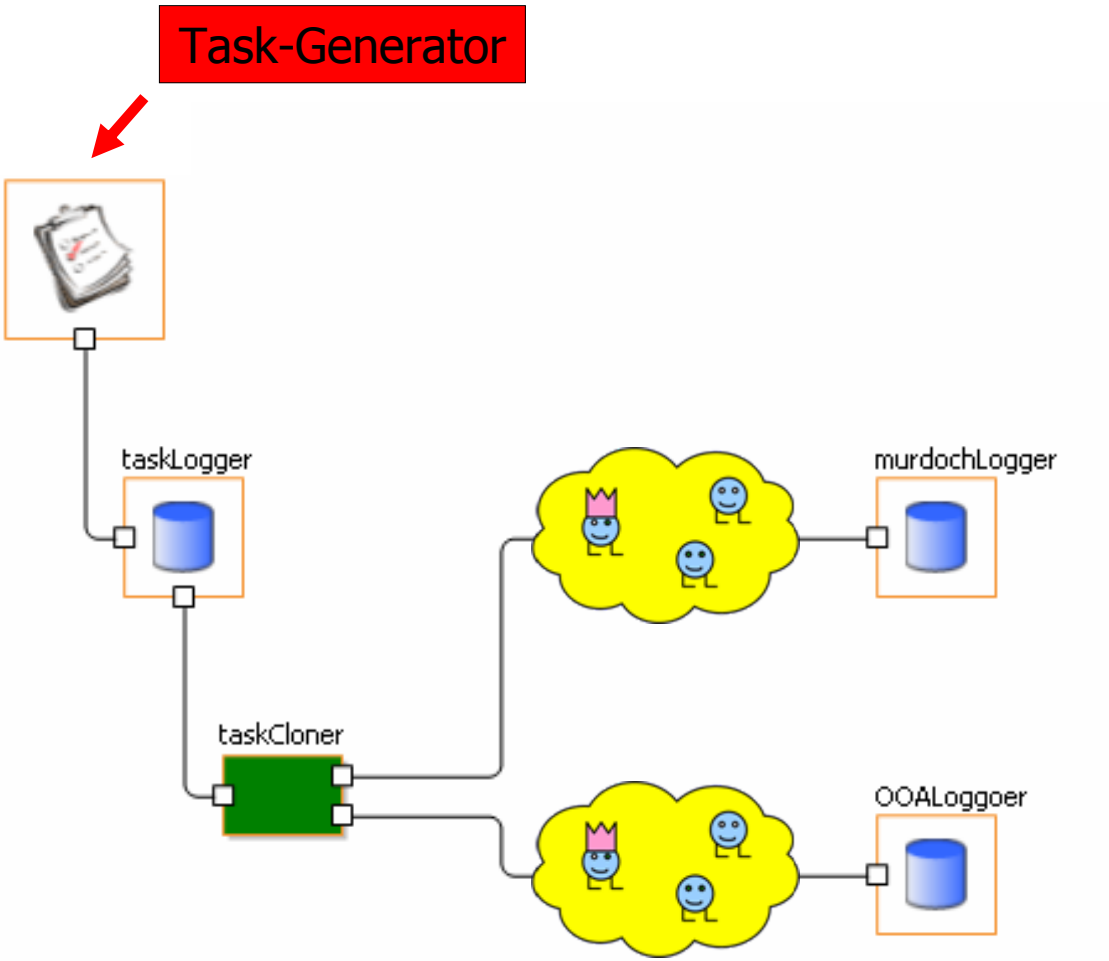
Umsetzung – Simulation:



- Anylogic
  - Simulationstool
  - XJ Technologies Company Ltd.
  - www.xjtek.com
  - Basiert auf Statecharts

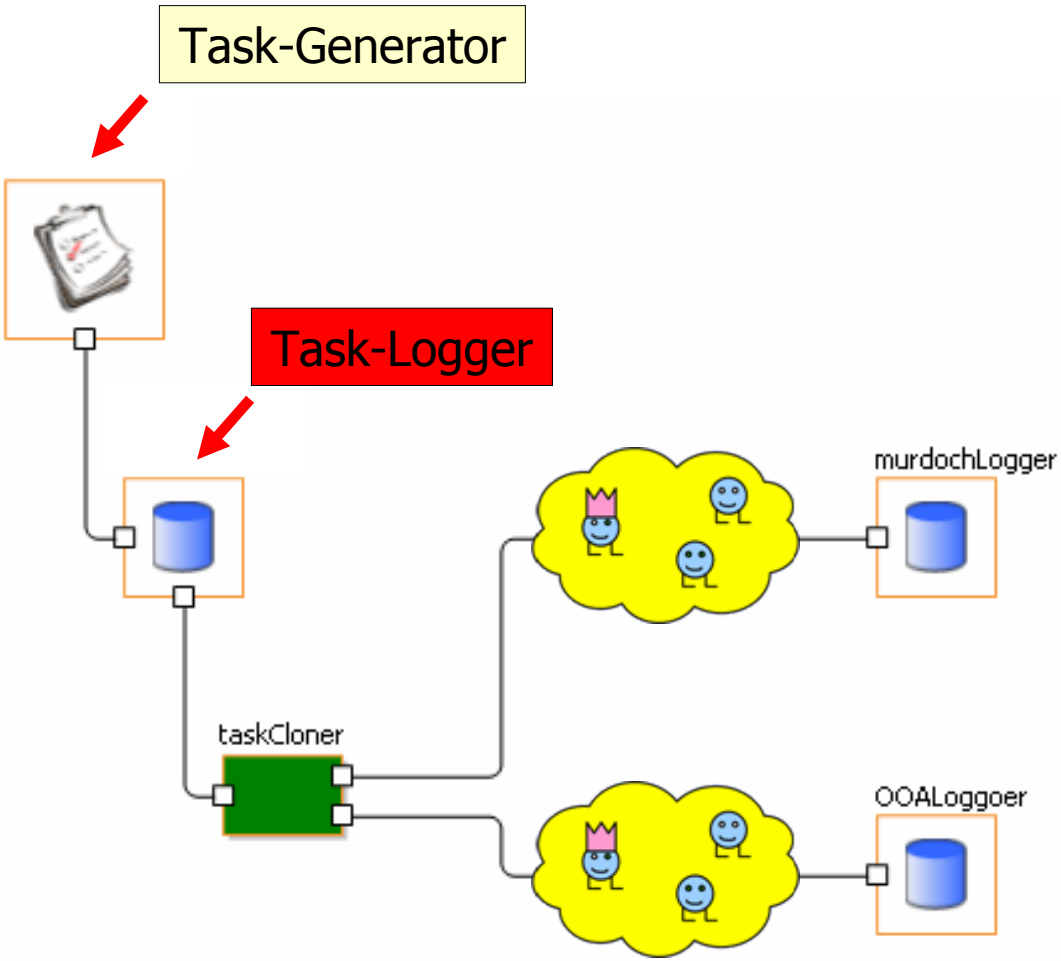
➔**Programmierung in Java**

Umsetzung – Simulation:



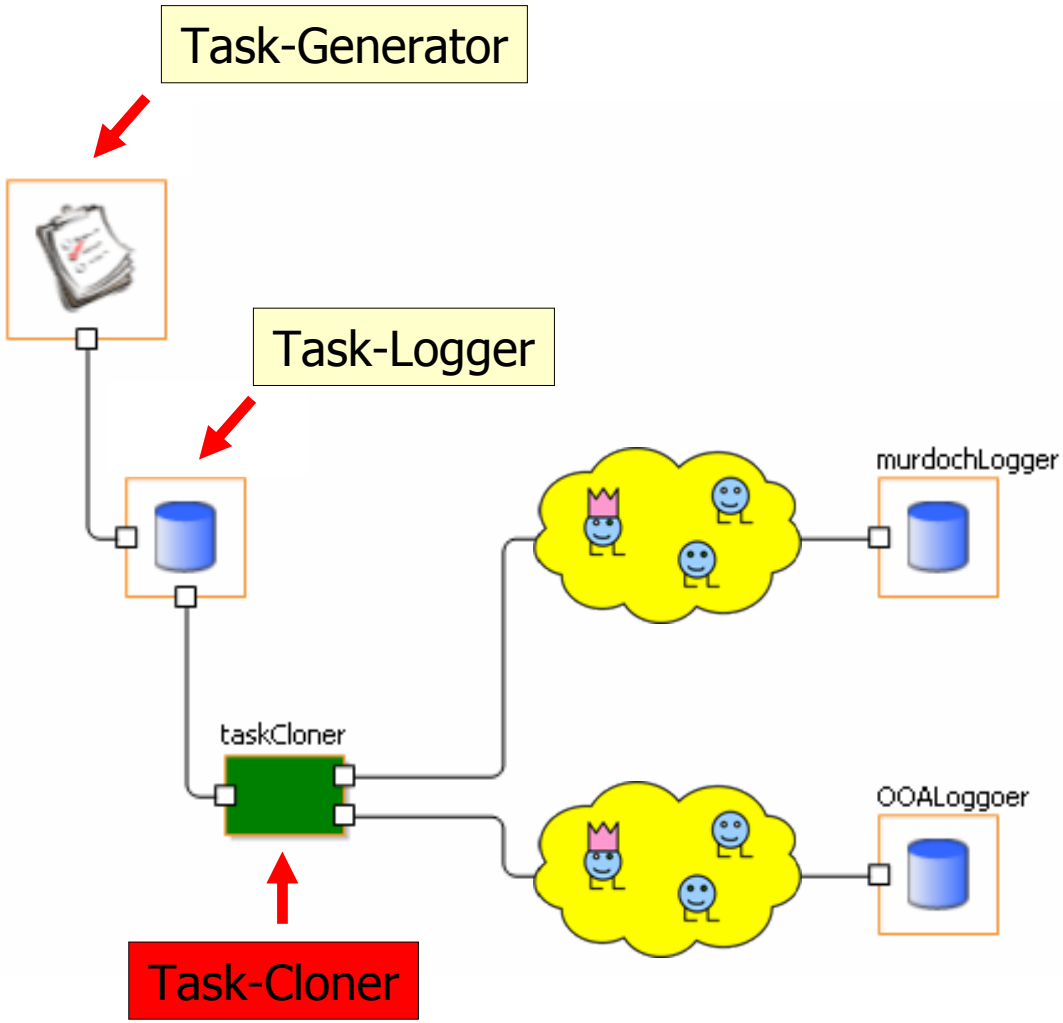
- Task-Generator
  - Erzeugen von Aufgaben
  - Zufällig oder Input-Datei

Umsetzung – Simulation:



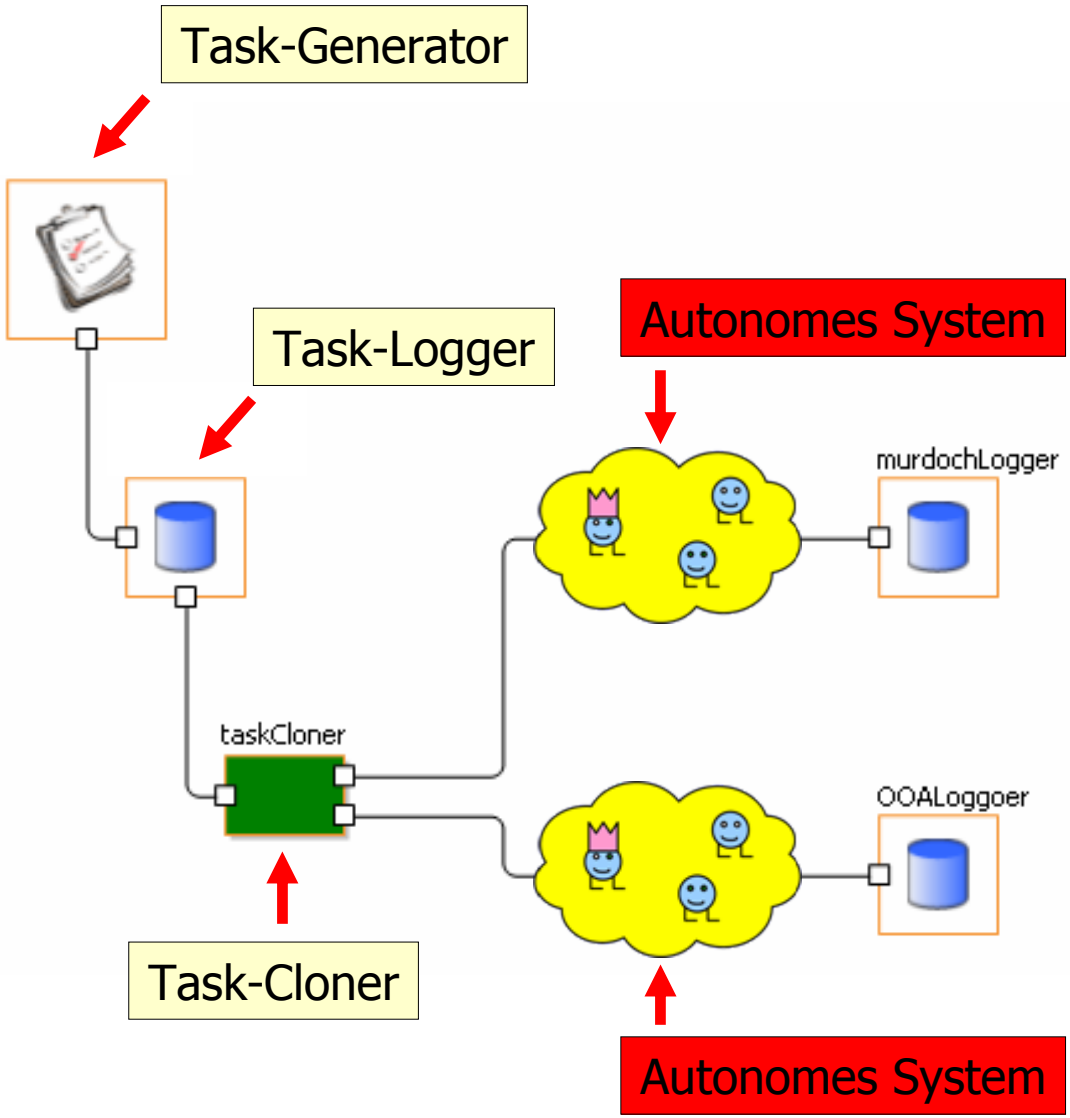
- Task-Generator
  - Erzeugen von Aufgaben
  - Zufällig oder Input-Datei
- Task-Logger
  - Speichern der Abfolge der Aufgaben
  - Input für Experiment

Umsetzung – Simulation:



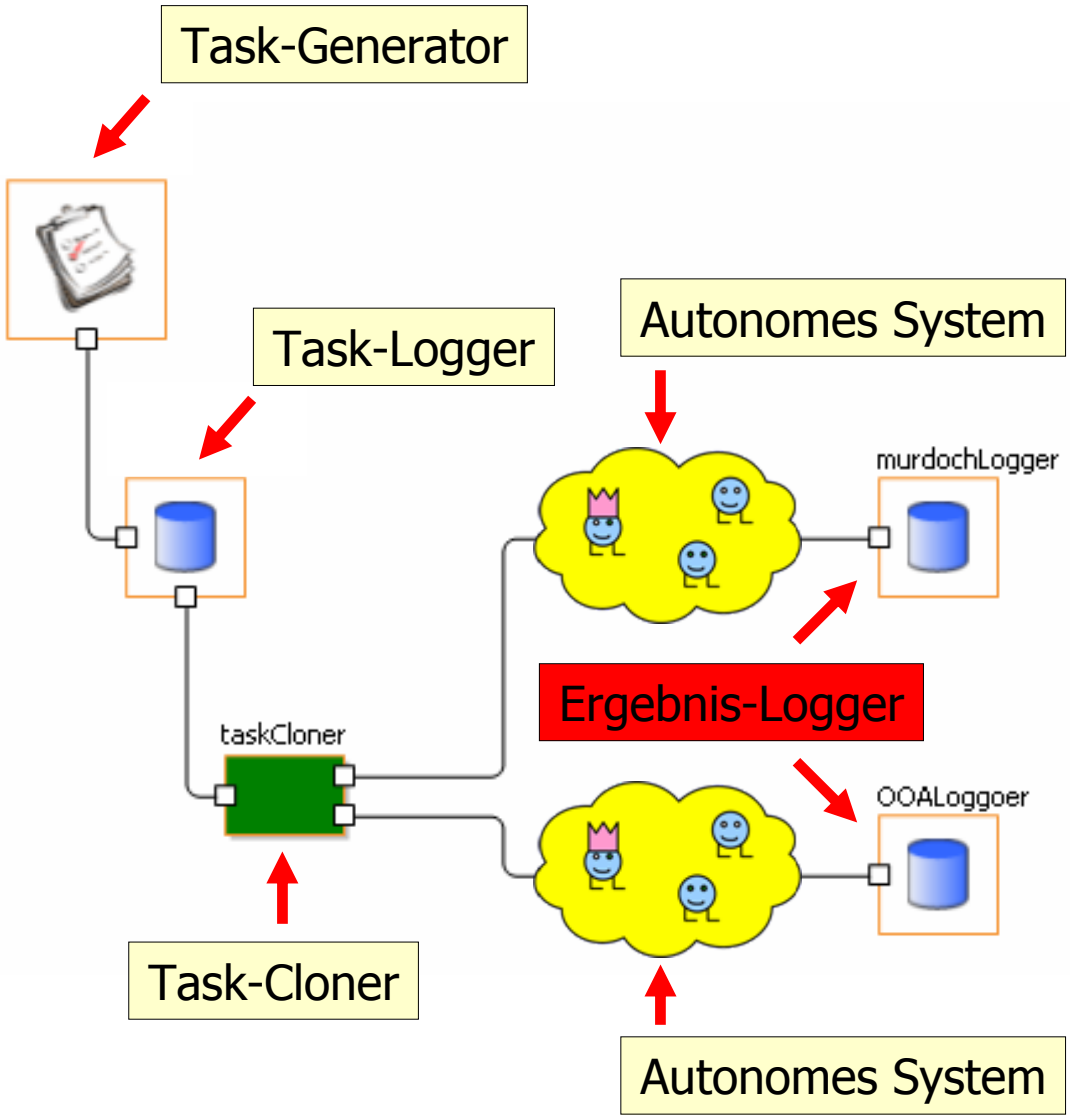
- Task-Generator
  - Erzeugen von Aufgaben
  - Zufällig oder Input-Datei
- Task-Logger
  - Speichern der Abfolge der Aufgaben
  - Input für Experiment
- Task-Cloner
  - Gleichzeitige Untersuchung mehrerer Mechanismen

Umsetzung – Simulation:



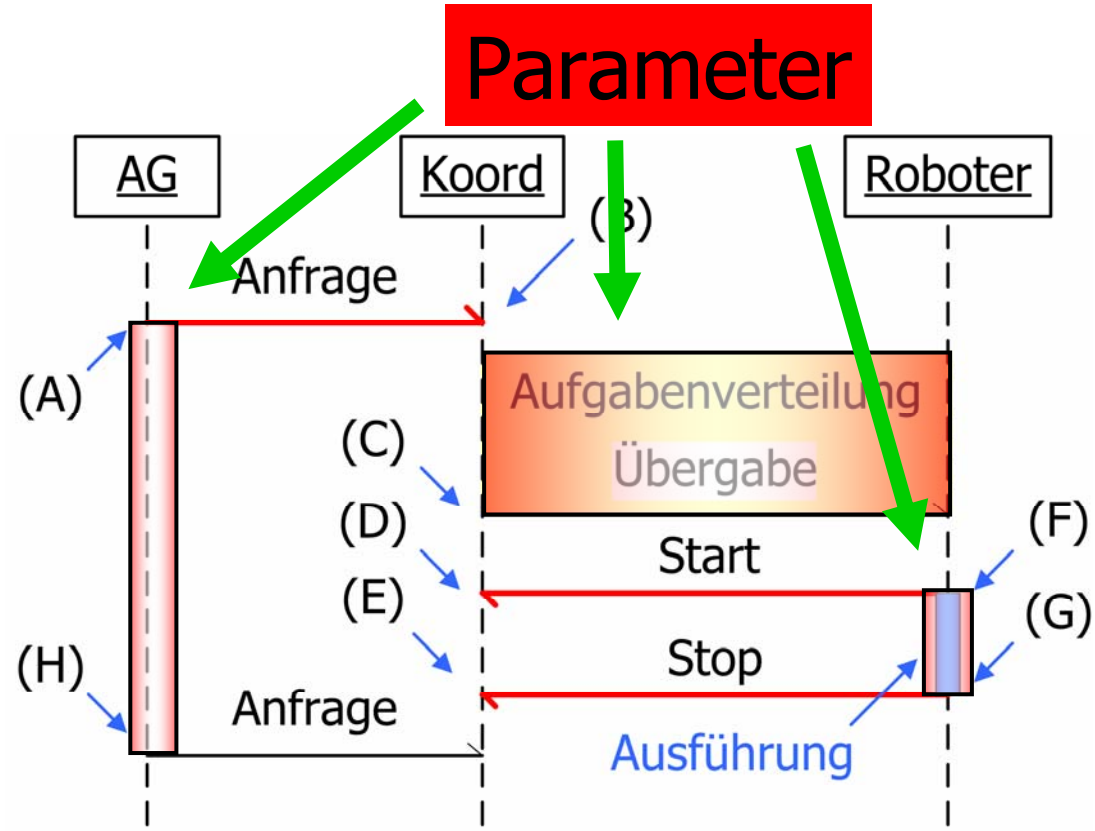
- Task-Generator
  - Erzeugen von Aufgaben
  - Zufällig oder Input-Datei
- Task-Logger
  - Speichern der Abfolge der Aufgaben
  - Input für Experiment
- Task-Cloner
  - Gleichzeitige Untersuchung mehrerer Mechanismen
- Autonomes System
  - Mechanismus

Umsetzung – Simulation:



- Task-Generator
  - Erzeugen von Aufgaben
  - Zufällig oder Input-Datei
- Task-Logger
  - Speichern der Abfolge der Aufgaben
  - Input für Experiment
- Task-Cloner
  - Gleichzeitige Untersuchung mehrerer Mechanismen
- Autonomes System
  - Mechanismus
- Ergebnis-Logger
  - Speichern der Ergebnisse

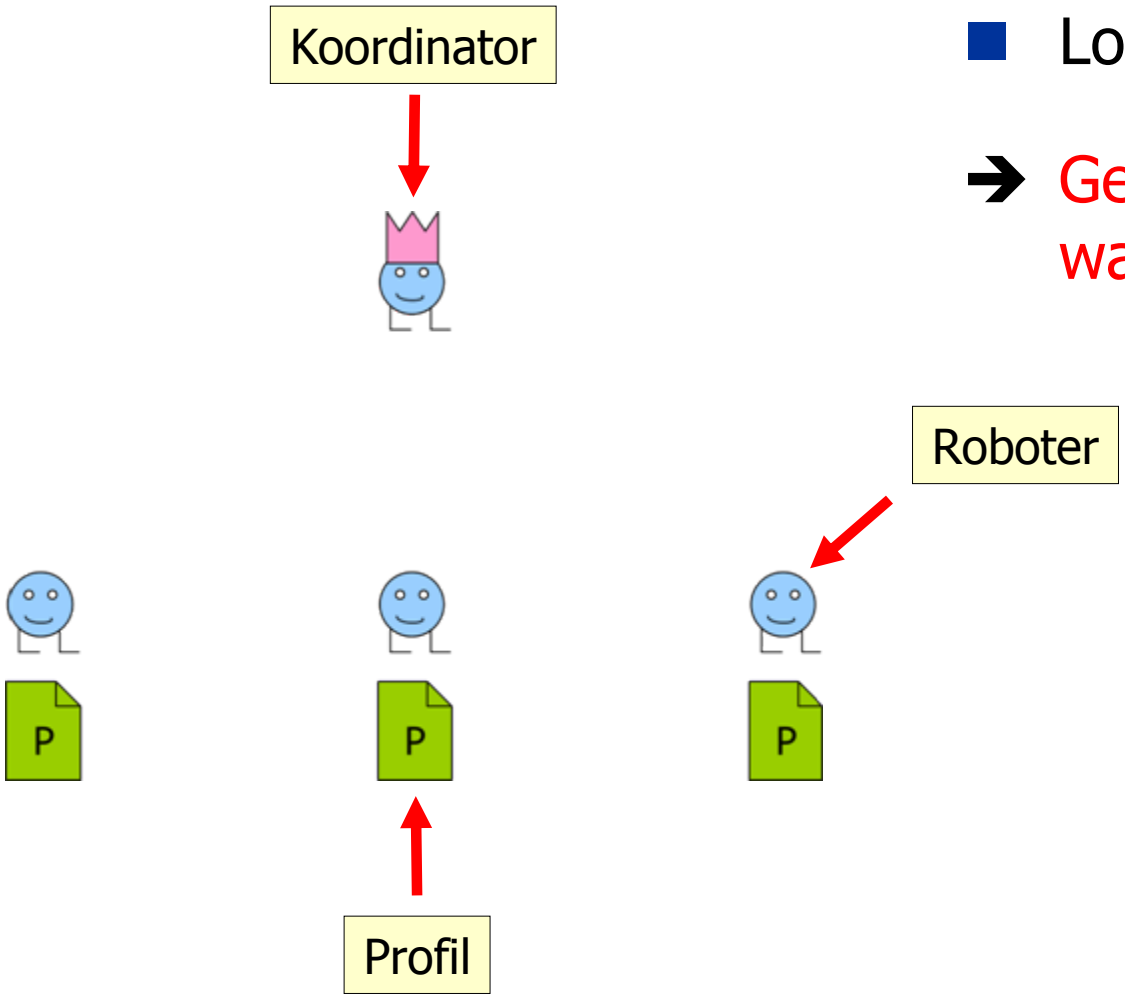
Beispiele:



AG: Aufgabengenerator
Koord: Koordinator

- Anfrageabstand
  - Zufällig
  - Konstant
- Mechanismen
  - MURDOCH
  - OOA
- Ausführungszeit
  - Wird berechnet
  - Input: Anfrage

Beispiele – MURDOCH:

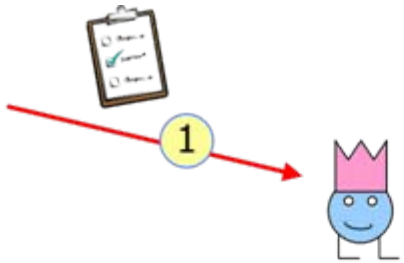


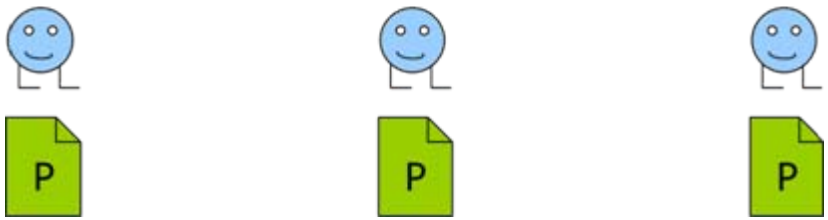
- Lokale Profilveraltung

➔ Geringer Koordinationsaufwand!

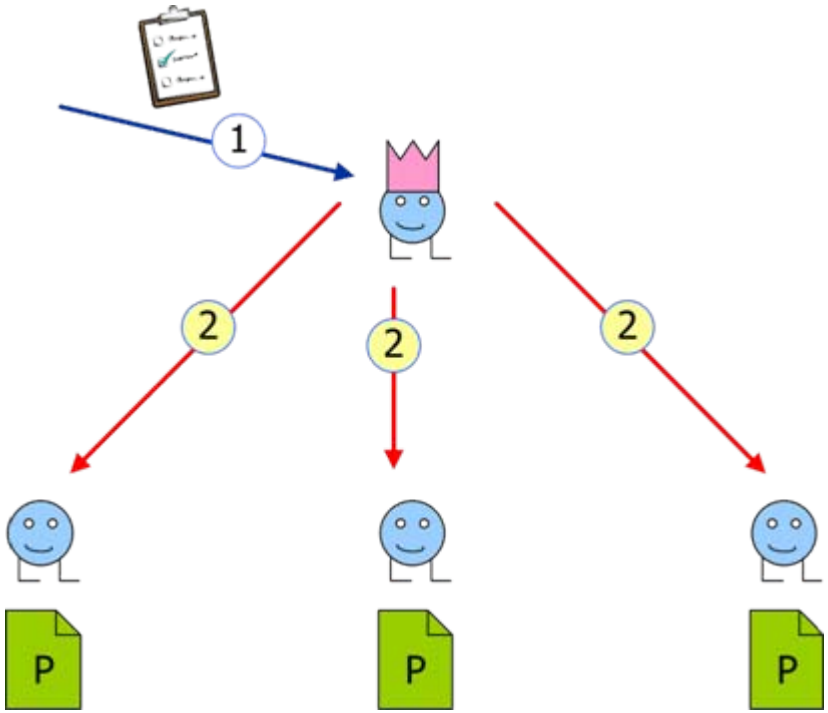
Beispiele – MURDOCH:

1) Ankunft der Aufgabe.



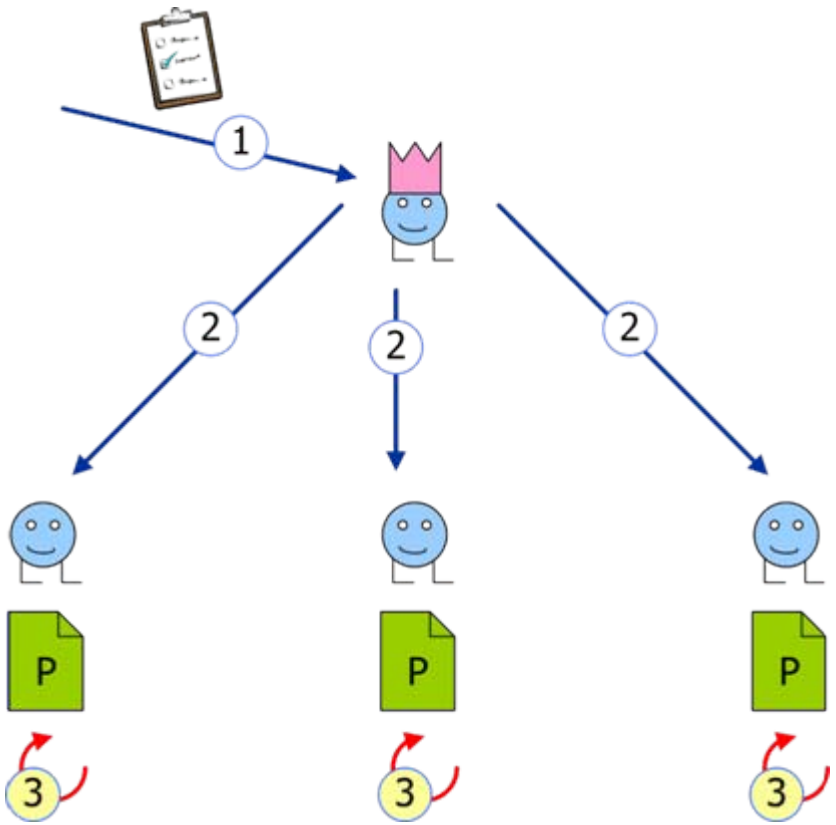


Beispiele – MURDOCH:



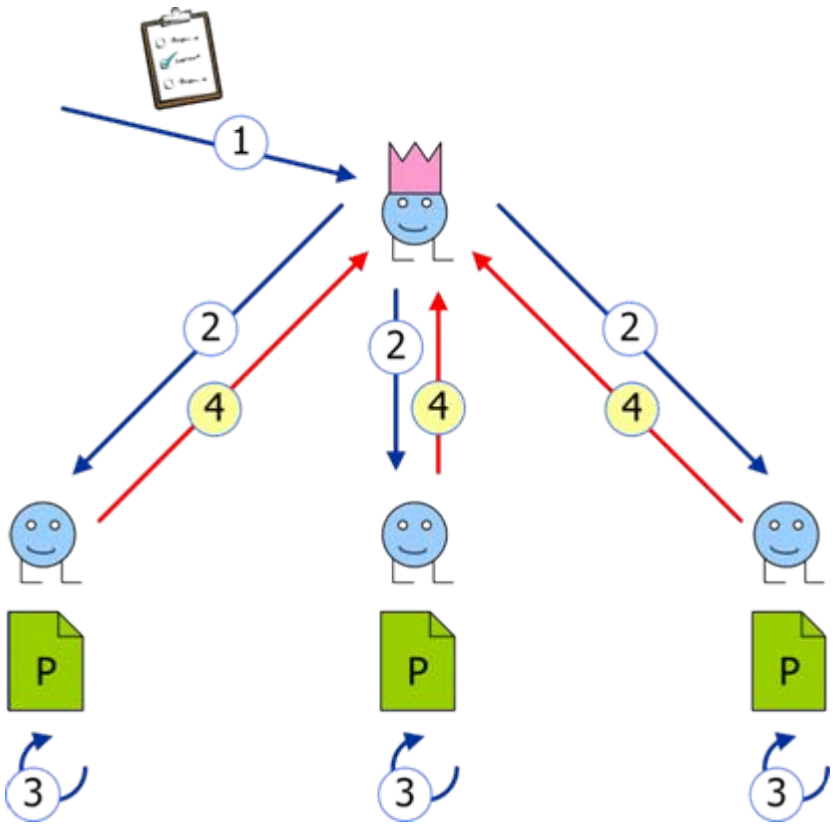
1) Ankunft der Aufgabe.
2) Broadcast an alle Roboter (*task announcement*).

Beispiele – MURDOCH:



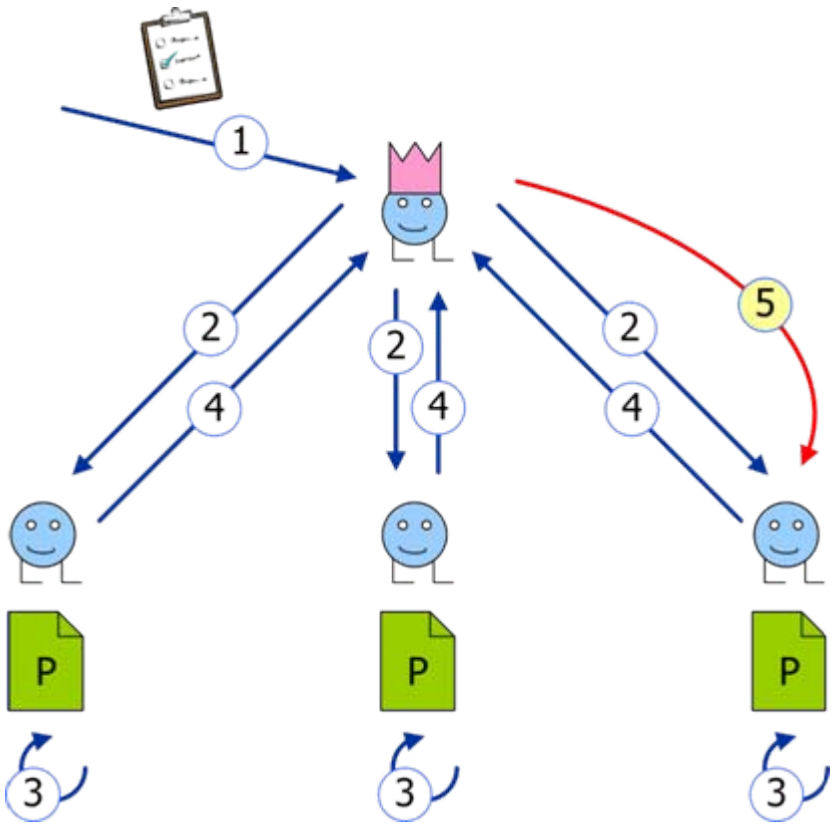
1) Ankunft der Aufgabe.
2) Broadcast an alle Roboter (*task announcement*).
3) Berechung der lokalen Fitness (*metric evaluation*).

Beispiele – MURDOCH:



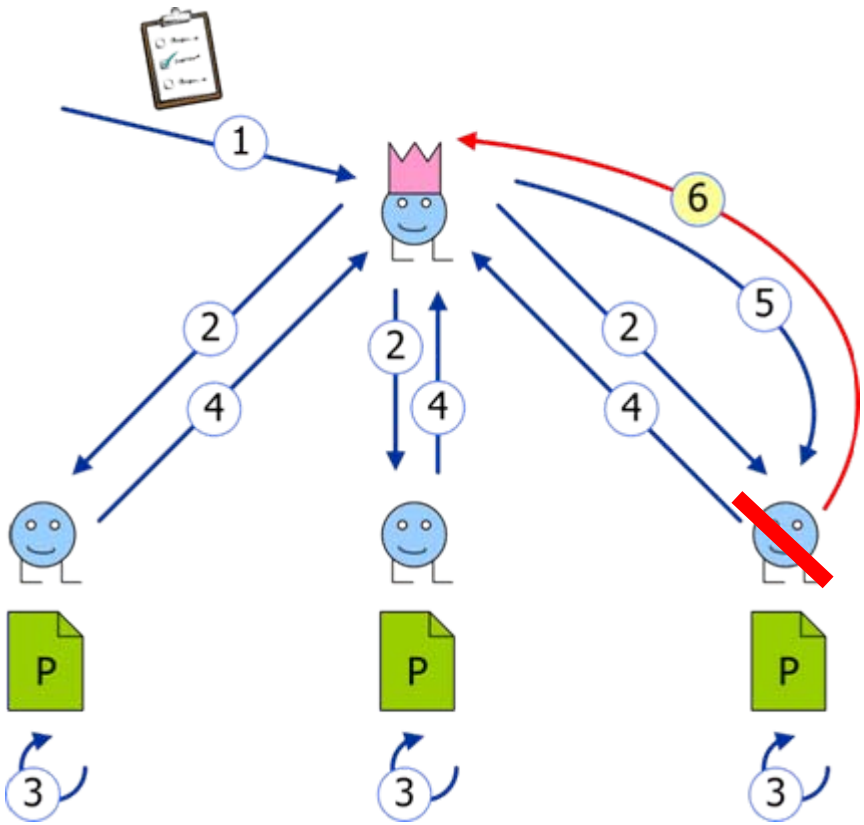
1) Ankunft der Aufgabe.
2) Broadcast an alle Roboter (*task announcement*).
3) Berechung der lokalen Fitness (*metric evaluation*).
4) Abgabe der Gebote (*bid submission*).

Beispiele – MURDOCH:



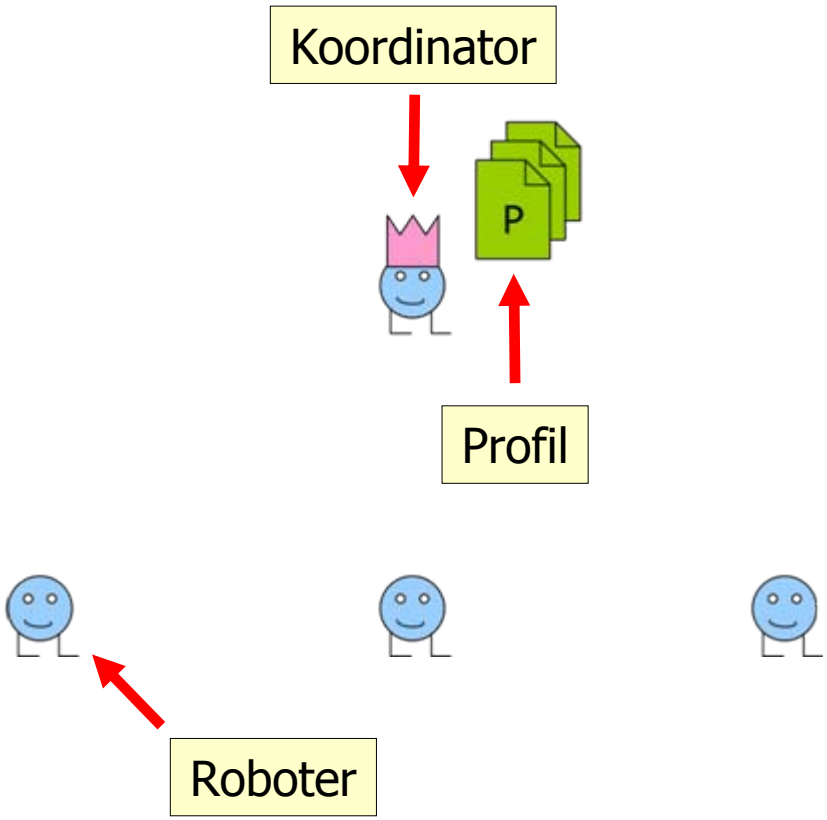
1) Ankunft der Aufgabe.
2) Broadcast an alle Roboter (*task announcement*).
3) Berechung der lokalen Fitness (*metric evaluation*).
4) Abgabe der Gebote (*bid submission*).
5) Ermittlung und Benachrichtigung des Gewinners (*close of auction*).

Beispiele – MURDOCH:



1) Ankunft der Aufgabe.
2) Broadcast an alle Roboter (*task announcement*).
3) Berechung der lokalen Fitness (*metric evaluation*).
4) Abgabe der Gebote (*bid submission*).
5) Ermittlung und Benachrich-tigung des Gewinners (*close of auction*).
6) Aufgabenausführung und Benachrichtigung des Koordinators (Start, Ende).

Beispiele – Open Agent Architecture OAA:

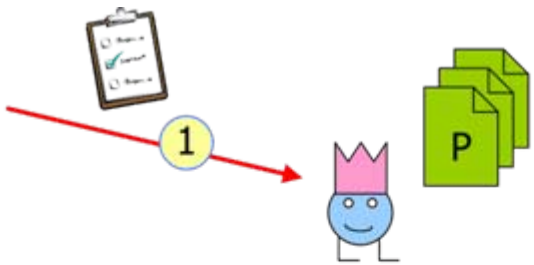


- Zentrale Profilveraltung

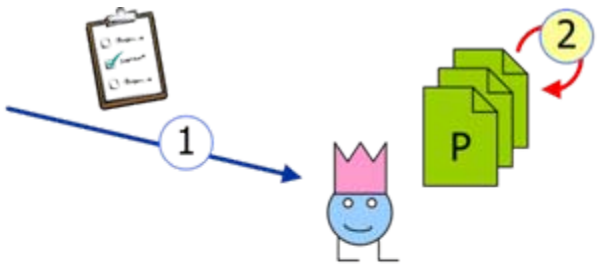
➔ Koordinationsaufwand, Statusverwaltung nötig.

Beispiele – Open Agent Architecture OAA:

1) Ankunft der Aufgabe.



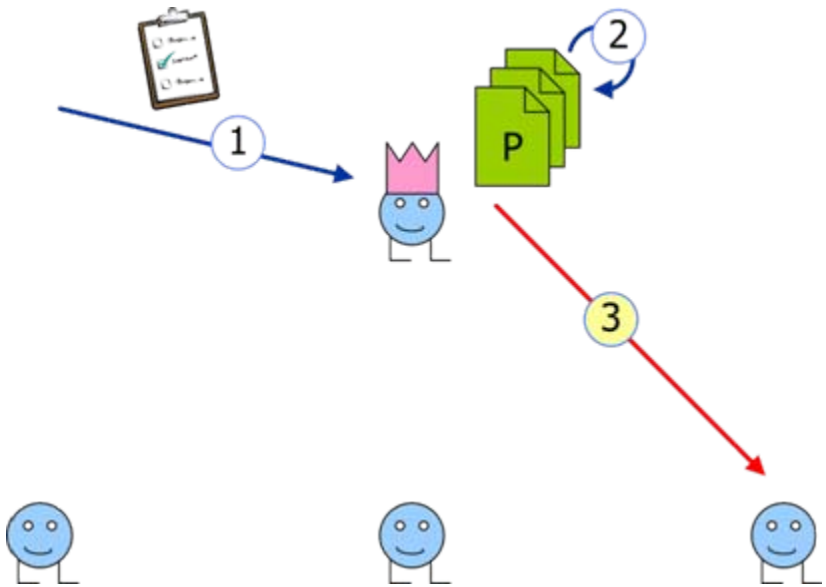
Beispiele – Open Agent Architecture OAA:



1) Ankunft der Aufgabe.

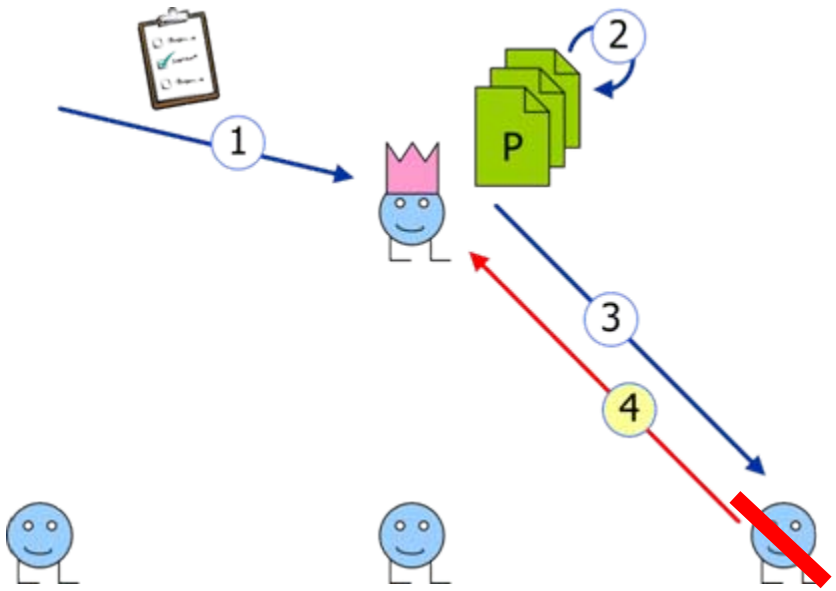
2) Ermittlung des am besten geeigneten Roboters durch den Koordinator (Profile).

Beispiele – Open Agent Architecture OAA:



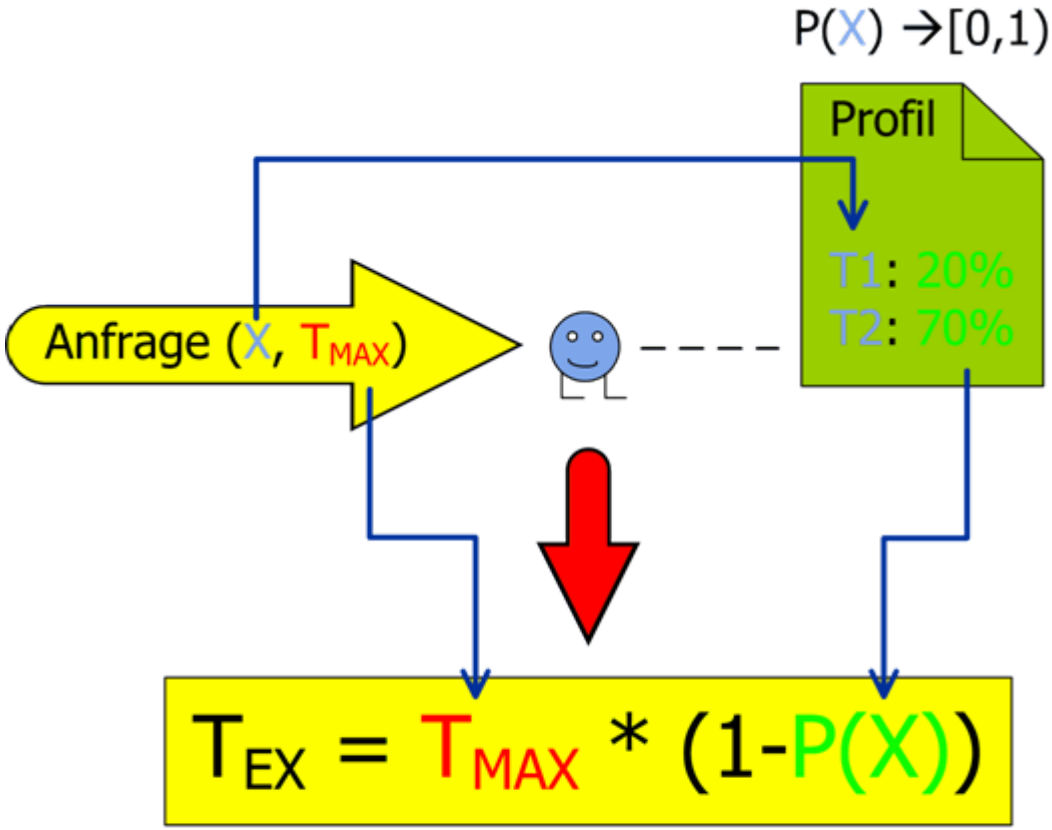
1) Ankunft der Aufgabe.
2) Ermittlung des am besten geeigneten Roboters durch den Koordinator (Profile).
3) Übergabe der Aufgabe an den Gewinner.

Beispiele – Open Agent Architecture OAA:



1) Ankunft der Aufgabe.
2) Ermittlung des am besten geeigneten Roboters durch den Koordinator (Profile).
3) Übergabe der Aufgabe an den Gewinner
4) Aufgabenausführung und Benachrichtigung des Koordinators (Start, Ende).

Beispiele – Ausführungszeit:

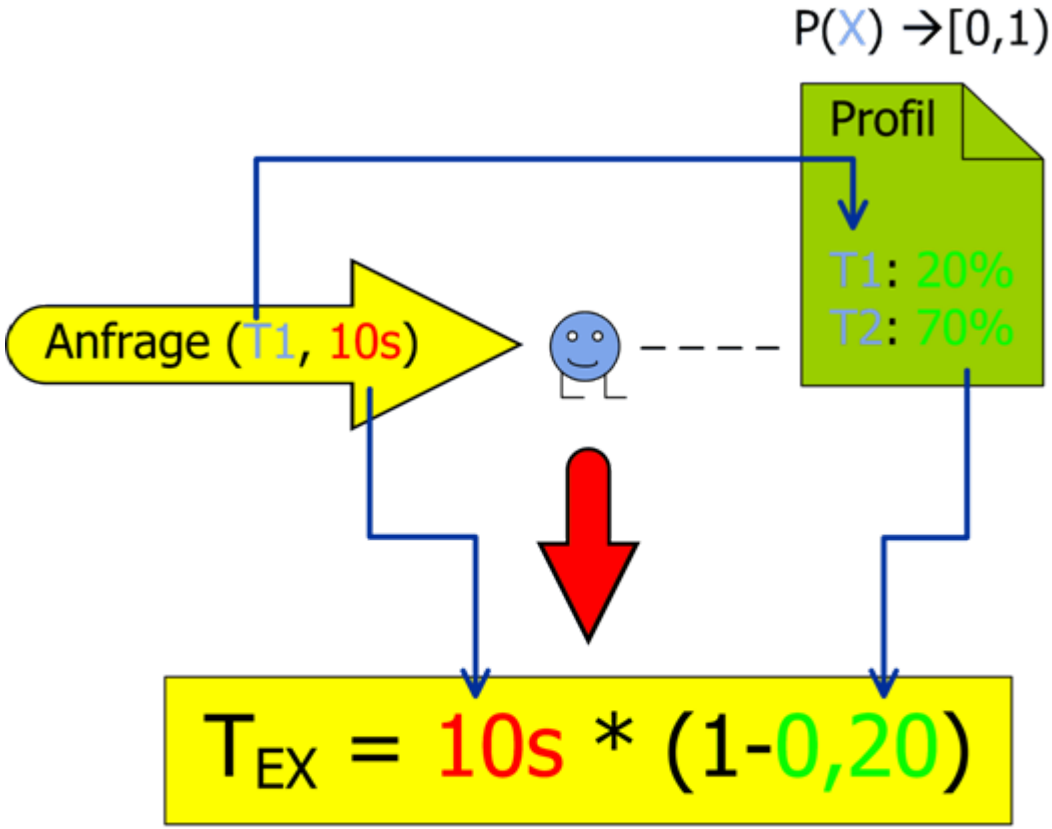


T_{EX} : Ausführungszeit
T_{MAX} : maximale Ausführungszeit
X: Aufgabentyp

- Profil
  - Für jeden Roboter
  - Was kann der Roboter?
  - 0: schlecht 1: gut
- Anfrage
  - Was ist zu tun?
  - Wie lange maximal?
- Ausführungszeit
  - Zeit, die benötigt wird
  - Roboter steht so lange nicht mehr zur Verfügung

➔ **Ranking möglich!**

Beispiele – Ausführungszeit:

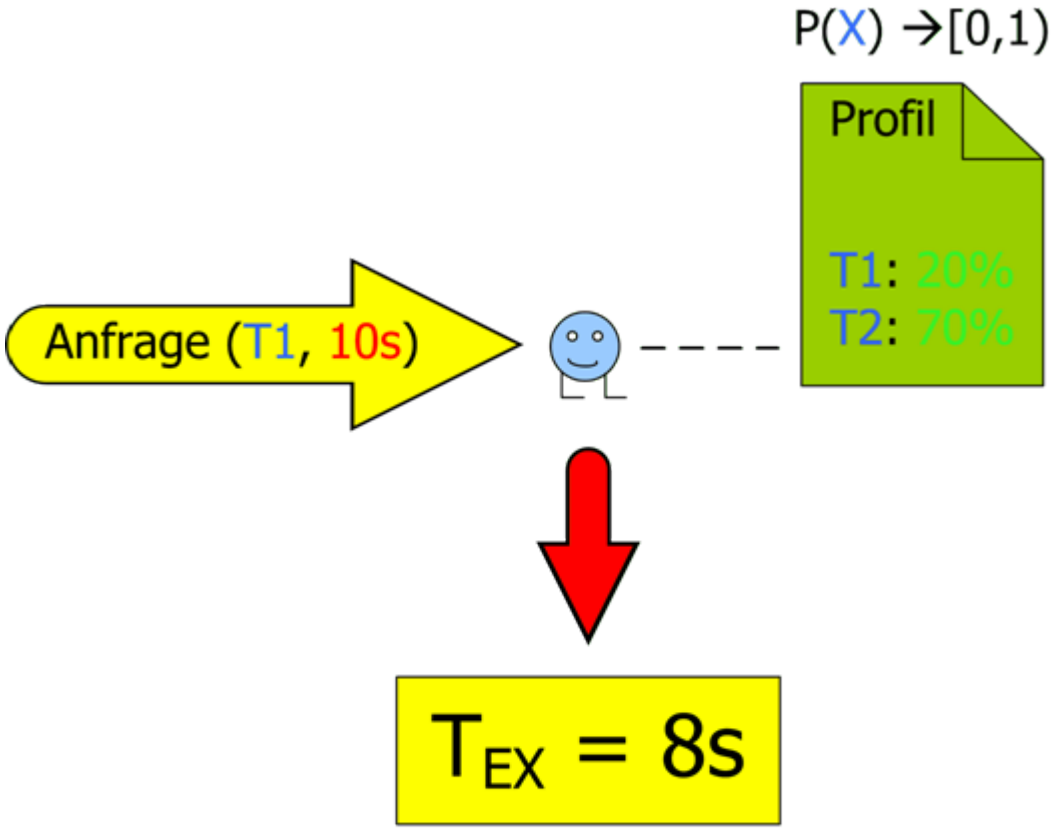


T_{EX} : Ausführungszeit
T_{MAX} : maximale Ausführungszeit
X: Aufgabentyp

- Profil
  - Für jeden Roboter
  - Was kann der Roboter?
  - 0: schlecht 1: gut
- Anfrage
  - Was ist zu tun?
  - Wie lange maximal?
- Ausführungszeit
  - Zeit, die benötigt wird
  - Roboter steht so lange nicht mehr zur Verfügung

➔ **Ranking möglich!**

Beispiele – Ausführungszeit:

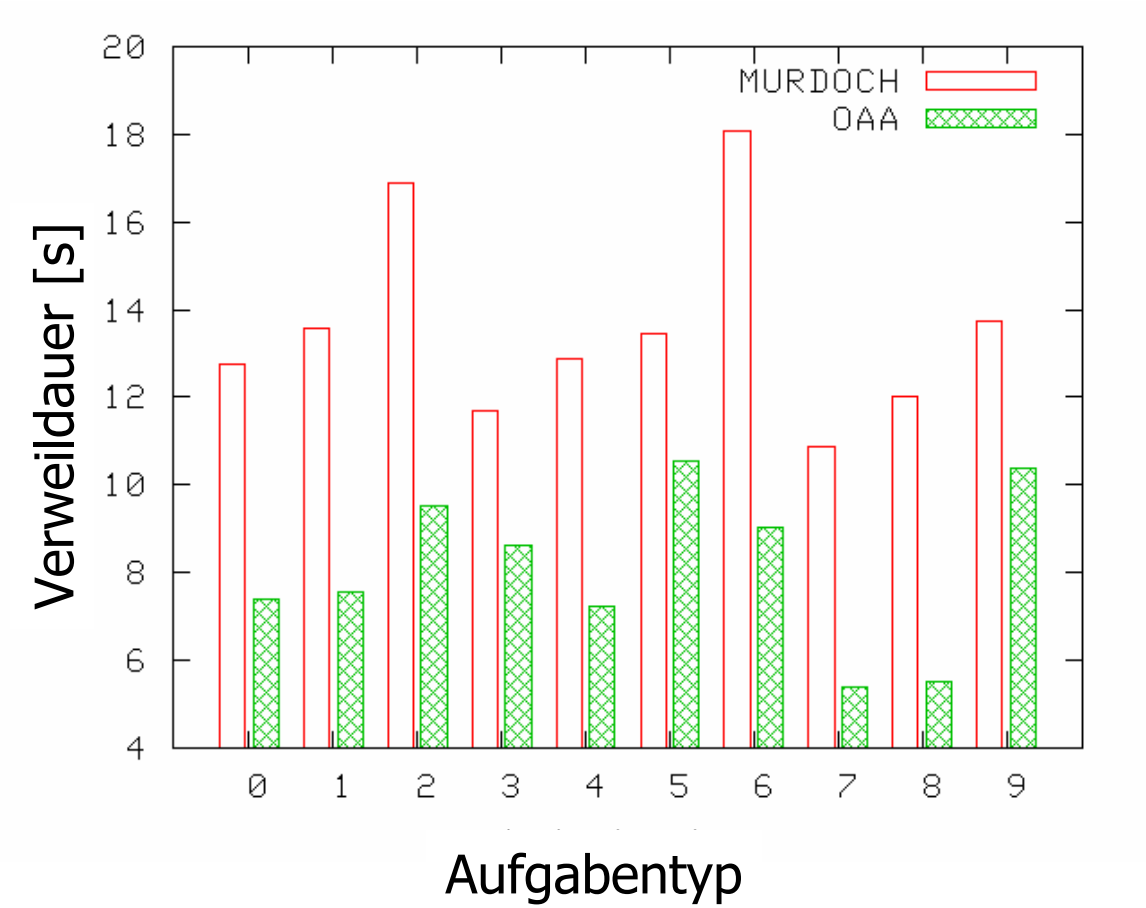


T_{EX} : Ausführungszeit
T_{MAX} : maximale Ausführungszeit
X: Aufgabentyp

- Profil
  - Für jeden Roboter
  - Was kann der Roboter?
  - 0: schlecht 1: gut
- Anfrage
  - Was ist zu tun?
  - Wie lange maximal?
- Ausführungszeit
  - Zeit, die benötigt wird
  - Roboter steht so lange nicht mehr zur Verfügung

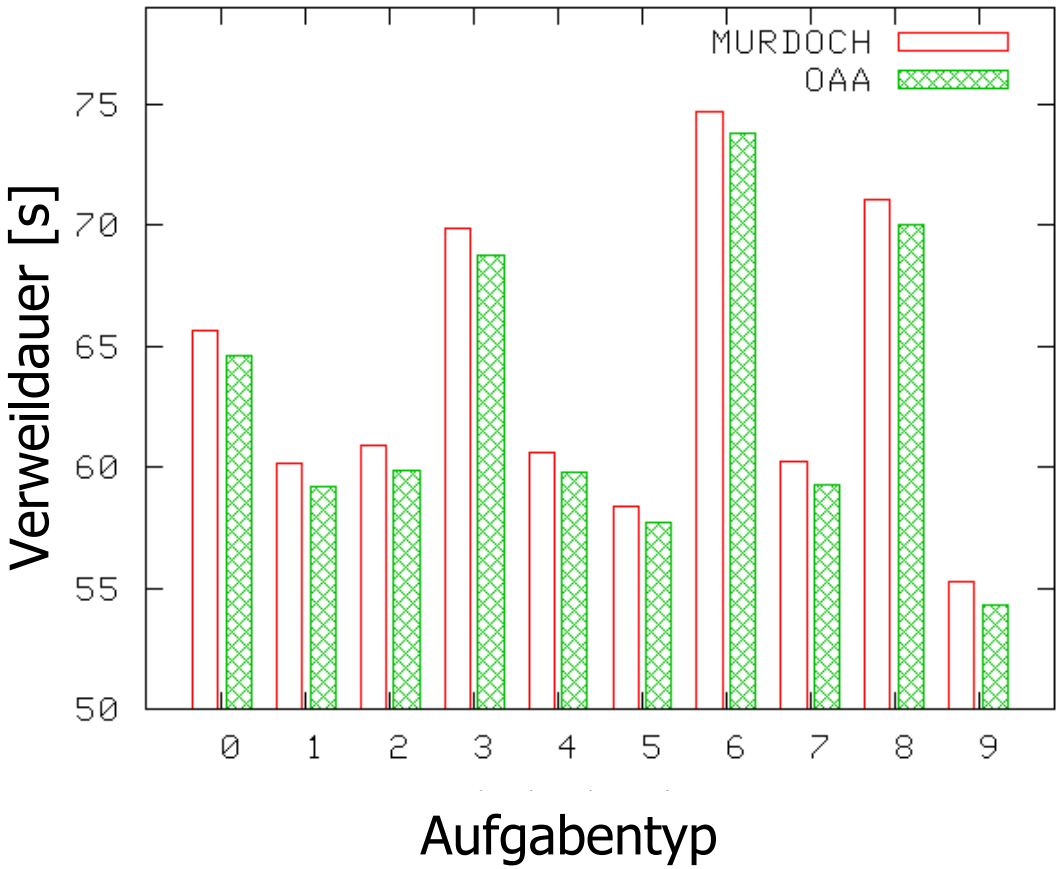
➔ **Ranking möglich!**

Beispiele – Ergebnisse:



- **Experimentelle Untersuchung - Verweildauer** der Aufgaben im System
- 3 Roboter; T_{ANF} = 2s; T_{MAX} = 11s
- Anzahl der Anfragen = 200; Auktionszeit von Murdoch = 1s)

Beispiele – Ergebnisse:



- **Simulation - Verweildauer** der Aufgaben im System
- 100 Roboter; T_{ANF} = 2s; T_{MAX} ϵ [100, 900] – für jede Anfrage zufällig ausgewählt
- Anzahl der Anfragen = 4500; Auktionszeit von Murdoch = 1s

Zusammenfassung / Ausblick:

- Einfaches Untersuchungsverfahren zur Bestimmung der Aufgabenverteilzeit und Verweildauer von Aufgaben in autonomen Systemen
- Ermöglicht Vergleich / Bewertung von Mechanismen
- Universell einsetzbar (Mechanismen sind austauschbar)
- Keine Zeit-Synchronisation der Knoten nötig
- Erprobt in Simulation und Experiment

➔ **Grundlage für eine Untersuchung von eigenen Mechanismen in selbst-organisierenden autonomen Systemen in Bezug auf ihre Echtzeitfähigkeit!**

FRAGEN?

Gerhard Fuchs, Falko Dressler

gerhard.fuchs@informatik.uni-erlangen.de

PEARL 2005

01.12.2005