# Hauptseminar

# Mobile Agenten - Ara

*Betreuer: Fritz Hohl*

*Bearbeiter: Michael Böser*

Universität Stuttgart
Institut für Paralelle und
verteilte Höchstleistungsrechner
WS 96/97

**1 EINLEITUNG ..... 3**

**2 ÜBERBLICK ÜBER DAS ARA - SYSTEM ..... 4**

2.1 KONZEPTE FÜR MOBILE AGENTEN IN ARA ..... 4
2.2 DIE ARA SYSTEM ARCHITEKTUR ..... 5
*2.2.1 Das Betriebssystem ..... 5*
*2.2.2 Der Ara-Kern ..... 5*
*2.2.3 Interpreter ..... 6*
*2.2.4 Agenten ..... 6*

**3 KOMMUNIKATION ..... 8**

3.1 KOMMUNIKATIONSKONZEPT ..... 8
3.2 NAMENGEBUNG UND ADRESSEN ..... 8
3.3 MIGRATION ..... 9
3.4 DIENSTE & TRADING ..... 9

**4 SICHERHEITSKONZEPTE ..... 10**

4.1 SICHERHEIT ..... 10
4.2 FEHLERBEHANDLUNG ..... 10

**5 SPRACHEN UND INTERAKTION ..... 11**

5.1 SPRACHEN ..... 11
5.2 INTERAKTION ..... 11

**6 BEISPIELANWENDUNG IN TCL ..... 12**

**7 GEPLANTE SYSTEMERWEITERUNGEN ..... 13**

# 1 Einleitung

Das Ara Projekt wird von Prof. Dr. J. Nehmer geleitet. Wissenschaftliche Mitarbeiter sind Dipl. Inf. H. Peine und Dipl. Inf. T. Stolpmann. Als externer Berater dient Dr, Peter Buhler (IBM Zürich).

Die Zielsetzung des Projekts ist es eine Betriebssystemunterstützung für Mobilcomputer zu schaffen. Die Nachteile der Mobilcomputer: Geringe Bandbreite, unzuverlässige Verbindung, niedrige Verfügbarkeit durch die Batterie und dynamische heterogene Umgebung, sollen durch eine Entkopplung von mobilem Client und Server durch ein Agentensystem verringert werden.

Da das System noch in Arbeit ist und die Quellen die verwendet wurden ein Ara - System Zugrundelegen das nicht dem entsprechenden Entwicklungsstand entspricht, kann es zu Unterschieden zwischen dem hier vorgestellten und dem endgültigen System kommen.

# 2 Überblick über das Ara - System

In diesem Kapitel werden Die Konzepte für mobile Agenten sowie die Systemarchitektur von Ara vorgestellt.

## 2.1 Konzepte für mobile Agenten in Ara

Da die Definitionen der einzelnen Begriffe nicht vorhanden bzw nicht klar sind werden sie nocheinmal kurz erklärt.

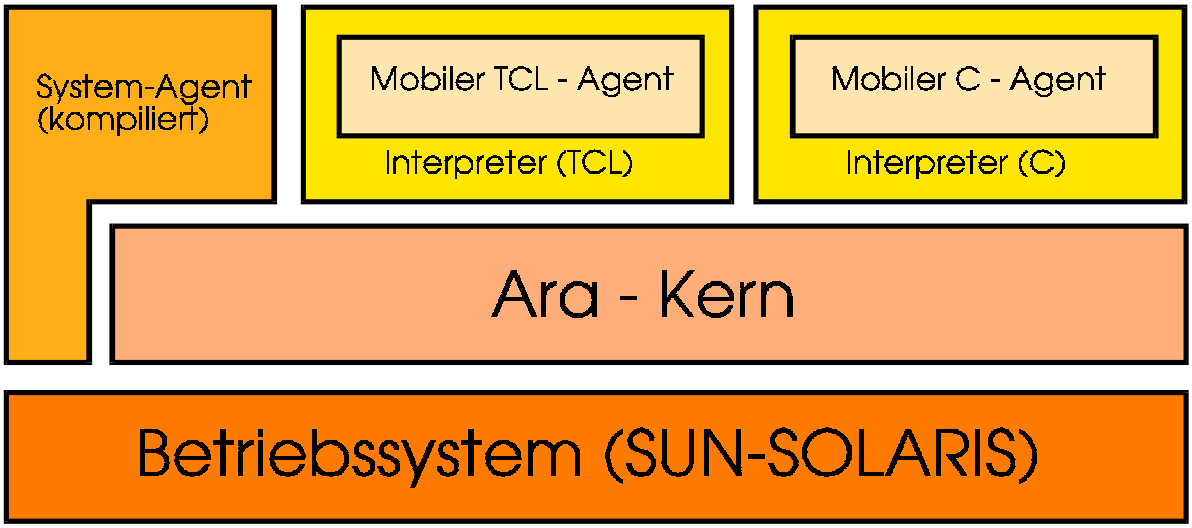
- **Agent**: Ein Agent in Ara ist ein in einer interpretierten Sprache ausgeführtes Programm, das die Fähigkeit besitzt frei zu migrieren und mit anderen Agenten zu interagieren. Einzige Ausnahme bilden die System-Agenten, die zwar schneller als die interpretierten Agenten sind aber auf einen Ort festgelegt sind und keine Sicherheit bezüglich des Hostsystems bieten.

- **Migration**: Übertragen eines ablaufenden Agenten von einem Rechner auf einen weiteren Rechner ohne Abbruch der Programmtätigkeit. Es wird also im weitesten Sinne ein Prozeß und dessen Zustand übertragen.

- **Dienst**: Ein Dienst wird von einem mobilen Agenten eingerichtet. Dieser Agent wird dann zum Server-Agenten. An den Server-Agenten können dann Anfragen bezüglich des Dienstes gerichtet werden. Der Dienst lebt nur solange wie der zugehörige Agent.

- **Ort**: Ein Ort kann Ziel einer Migration sein, ein Agent an einem Ort kann Dienste anbieten oder abrufen oder mit anderen Agenten interagieren.

## 2.2 Die Ara System Architektur




### 2.2.1 Das Betriebssystem

Als Betriebssystem wird zur Zeit das Unix der SUN-SOLARIS Maschinen verwendet. Es sind Portierungen auf andere Betriebssysteme bzw Hardwareplattformen geplant. Das Hostsystem stellt Funktionen wie Ein/Ausgabe, das Benutzerinterface, das Filesystem, externe Anwendungen und das Netzwerkinterface zur Verfügung. Da das Ara-System portabel und auch auch auf anderen Betriebssystemen laufen soll kann nur ein Teil der Funktionalität des Betriebssystems vorausgesetzt werden. Es ist Zwingend notwendig einen Einheitlichen Zugriff auf das Hostsystem zu schaffen, doch dazu mehr in der Beschreibung des Ara-Kerns.

### 2.2.2 Der Ara-Kern

Der Ara-Kern verwaltet die Agenten-Prozesse, vermittelt ihre Interaktion und ihr Zugriff auf das Hostsystem.

Das **Hostsystem** wird über ein sogenanntes Hostinterface angesprochen das grundlegende Systemfunktionen vereinheitlicht und vom Hostsystem unabhängig macht.

Dazu gehören:

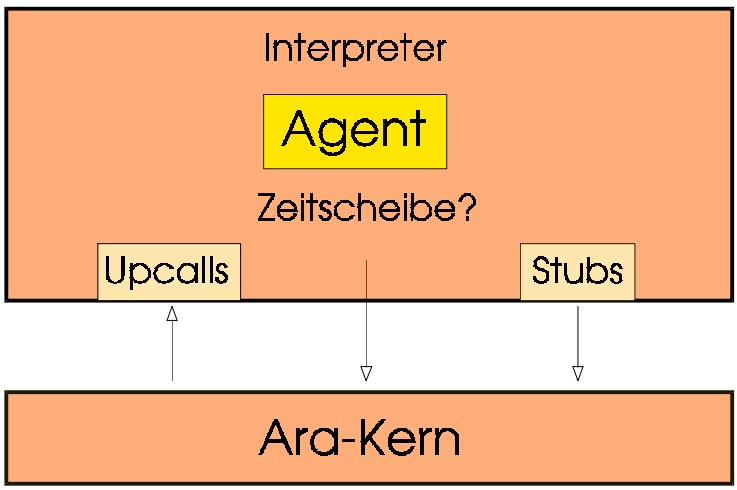
- Physische Ein- / Ausgabe (zur Zeit nur sprachspezifische Ein- / Ausgabe)

- Benutzer Interface (basiert auf TK)
- Filesystem (Flat Filesystem mit Zugriffsrechte für Agenten)
- Netzwerk (Konzept der Kommunikationskanäle wird für Netzwerkzugriff und lokalen Zugriff unterstützt.)
- Interface zu externen Applikationen (Kommunikation über sogenannte Stellvertreter Agenten die eine Schnittstelle für Client Agenten herstellen und evtl die Daten in ein geeignetes Format umwandeln.)
- Zeitfunktionen (noch nicht verfügbar)

Diese Funktionalität findet man auch im Sprachunabhängigen **Programmierinterface** (API). Das API ist das Laufzeitsystem für die Interpreter.

### 2.2.3 Interpreter

Die Interpreter enthalten sogenannte „stubs“ d.h. Aufruf-Schnittstellen die festlegen welche Ara-Funktion mit dem entsprechenden Sprachbefehl (TCL, C) ausgeführt werden soll. Sie stellen auch sogenannte „upcalls“ d.h. Funktionen für das Interpreter Management zur Verfügung die dem Ara-Kern erlauben den Interpreter zu verwalten. Außerdem werden Funktionen zur Speicherplatzsteuerung und Prozeßmanagement zur Verfügung gestellt doch dazu später mehr.



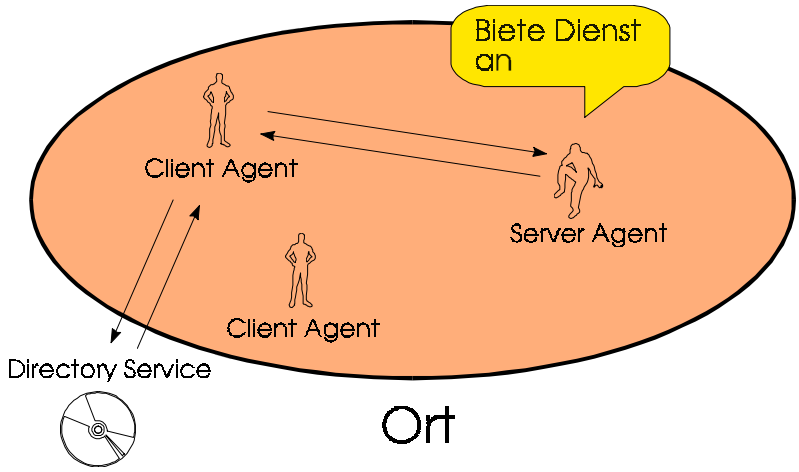

### 2.2.4 Agenten

Die Agenten sind entweder in TCL oder in C geschrieben. Für System-Agenten steht zur Zeit nur C zu Verfügung. Agenten bestehen aus einem normalen Textfile das von dem jeweiligen Interpreter bearbeitet wird. Solche Agenten können nur über das Programmierinterface auf das Hostsystem zugreifen. System-Agenten werden direkt ausgeführt und können anders als die

interpretierten Agenten direkt auf das Betriebssystem zugreifen. Sie unterliegen Beschränkungen was Migration und Sicherheit angeht.

# 3 Kommunikation

Hier werden die Aspekte Kommunikationskonzept, Namengebung und Adressen, Migration und Trading im Zusammenhang mit Ara erläutert.




## 3.1 Kommunikationskonzept

Das Ara-System ist auf lokale Interaktion zwischen Agenten ausgelegt, die globale Kommunikation wird aber nicht grundsätzlich verboten. Ara verwendet eine Art Client/Server Interaktion für den Nachrichtenaustausch. Der Ara-Kern stellt dazu das Konzept der Orte zur Verfügung. Dies ist ein Treffpunkt für Agenten mit einem im System bekannten Namen. Agenten können an diesem Ort als Client und Server interagieren. Der Agent der einen Dienst bekannt gibt agiert als Server Agent. Client Agenten können Server Agenten besuchen und mit ihnen kommunizieren. Der Client Agent kann auch einen Dienst aufrufen, der noch nicht existiert und muß dann warten oder bekommt eine Fehlermeldung zurück. Der Server Agent merkt sich die Namen bzw ID der Client Agenten und kann mit den Client Agenten individuell kommunizieren.

## 3.2 Namengebung und Adressen

Agenten bekommen bei ihrer Erzeugung eine eindeutige Agenten ID (fortlaufende Nummer), die in einer Variablen gespeichert wird.

Dienste bekommen einen Namen von einem Agenten zugewiesen. Der Agent verwaltet den von ihm eingerichteten Dienst. Ein Agent wird zu einem Client- Agenten indem er den Dienst unter seinem Namen aufruft und kann dann mit dem Server- Agenten interagieren.

Orte werden als Objekte repräsentiert. Es gibt aber zusätzlich eine Zeichenstring-repräsentation die Applikationen benutzen können um Orte zu generieren. Diese besteht aus dem DNS Namen und dem TCP Port des Ziel Ara-Systems „default“ Ort.

## 3.3 Migration

Das Migrieren ist über den Systemkern mit einem Befehl möglich, der Kern erledigt dann den Rest. Intern wird der Agent mittels TCP-Protokoll an das Ara-System am gewünschten Ort übertragen und dann an der Stelle weiter ausgeführt wo er vor der Übertragung aufgehört hat. Der Agent wird komplett übertragen(Transparente Migration). Andere Systemobjekte wie Dienste, Files oder Fenster werden nicht mit übertragen.

## 3.4 Dienste & Trading

Es wird eine lokale Dienstvermittlung angeboten, den sogenannten „directory service“, in dem der Agent nach bestimmten Diensten suchen kann und anschließend an den Ort migrieren der diesen Dienst zu Verfügung stellt.

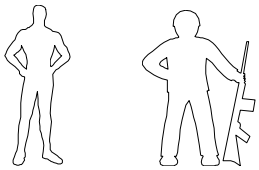
Wie wir schon erfahren haben gibt es Client und Server-Agenten wobei nur die Server-Agenten Dienste einrichten können indem sie dem Dienst einen Namen geben. Falls der Dienst nicht mehr an einem Ort zur Verfügung steht bekommt der Agent der den Dienst anfordert, je nach Befehl, eine Fehlermeldung oder er wartet bis der Dienst wieder angeboten wird.

Trading ist noch nicht implementiert, es gibt aber eine Möglichkeit den Ressourcenverbrauch (Zyklen, Hauptspeicher, Plattenplatz) für einen Agenten über sogenannte Ressourcenkonten zu begrenzen. Die Ressourcenbegrenzung kann global bei der Erstellung oder lokal von einem Ort festgelegt werden.

# 4 Sicherheitskonzepte

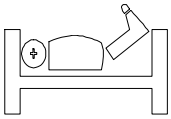
In diesem Kapitel werden kurz die Maßnahmen beschrieben die verhindern sollen das Agenten oder das Hostsystem manipuliert werden.

## 4.1 Sicherheit



Die Sicherheit zwischen den Agenten wird durch den Interpreter im Kern gewährleistet da Interpreter in der Regel volle Kontrolle über die von ihnen ausgeführten Programme haben.
Um die Beeinflussung des Agentensystems zu vermeiden wird jeder Agent in seinem eigenen Adressraum ausgeführt. Der Interpreter sorgt dafür das der Agent nur auf seinen eigenen Adressraum zugreifen kann. Um mit anderen Agenten oder dem Kern zu kommunizieren müssen Funktionen des Kerns benutzt werden. Der Kern und seine Objekte (andere Agenten, Dienste) liegen außerhalb des Adressbereichs des Agenten. Der Zugriff auf andere Agenten oder Dienste erfolgt indirekt über Identifikationen. Die absolute Adresse wird aus einer Objekttabelle, die der Kern verwaltet, gelesen. Es sind Orte mit Pförtnern geplant sowie eine Verschlüsselung der Agentenübertragung.

## 4.2 Fehlerbehandlung



Als Fehlerbehandlung wird das sogenannte „Checkpointing“ verwendet, daß es ermöglicht eine Kopie von einem Agenten an einem Ort auf persistente Medien zu sichern und nach bedarf wiederherzustellen (Migration auf Platte). Die vorherige Inkarnation wird nach dem Recovery gelöscht.

# 5 Sprachen und Interaktion

Das Kapitel behandelt die Sprachunterstützung und die Interaktion mit dem Benutzer.

## 5.1 Sprachen

Durch die Trennung von sprachspezifischen (im Interpreter) und sprachunabgängigen Funktionen (im Kern) sowie dem MACE-System ist es möglich neue Sprachen zur Verfügung zu stellen. Mit dem MACE-System ist es möglich Frontends des GNU-Compilerpackets zu verwenden wie z.B. C, C++, Objective C, Fortran, Ada und Pascal.

Es sind Interpreter in den Sprachen TCL, C und JAVA geplant. Realisiert wurde bisher nur der TCL und der C Interpreter. TCL wurde gewählt weil es eine schnelle Entwicklung zuläßt. Der C-Interpreter ist schneller in der Ausführung da der C-Quellcode in einen Mace-Bytecode vorcompiliert wird und nur der wesentlich kompaktere Bytecode interpretiert wird. Über den JAVA - Interpreter ist zur Zeit noch nichts bekannt.

Für System-Agenten steht bisher nur C als Programmiersprache zur Verfügung.

## 5.2 Interaktion

Es wird an einem Host Interface gearbeitet das kontrollierten Zugriff auf das Hostsystem zuläßt. Dieses ist ähnlich wie ein Programmier Interface und stellt Funktionen für den Zugriff auf das Filesystem, die grafische Oberfläche, Timer sowie das Netzwerk zu Verfügung.

Das graphische Benutzerinterface wird auf TK dem graphischen Toolkit von TCL basieren. Um die Sprachunabhängigkeit zu gewärleisten ist es geplant TK von TCL zu isolieren und es auch für andere Sprachen verfügbar zu machen.

Das Netzwerkinterface soll einen Treiber für Funkmodems enthalten. An den anderen Schnittstellen wird noch gearbeitet.

# 6 Beispielanwendung in TCL

### DateiSuchAgent:

```
ara_agent {
        set sp [ara_announce DateiSuch-Service]
        puts „DateiSuch-Service installiert.“
        set request [$sp fetch] # Anfrage einhohlen, falls keine anliegt warten ..
        set dateiname [lindex $request 0] # Request Format {daten client-name}
        while {[string compare $dateiname exit]} {
                suche Datei und lese Datei in einen String $DateiInhalt
                $sp reply [lindex $request 1] $DateiInhalt
                set request [$sp fetch]
                set item [lindex $request 0]
        }
        $sp renounce
        puts „DateiSuch-Service geschlossen.“
    }
ara_sleep 5;
```

### DateiServerAgent:

```
set sp [ara_meet DateiSuch-Service]
    if {$sp == „„} {
        puts „Kann DateiSuch-Service nicht finden!“
        exit
    }
    set DateiInhalt [$sp <Datei>]
    Datei ausgeben
$sp leave
```

# 7 Geplante Systemerweiterungen

- MACE System
- Sicheres und Portables Hostinterface
- Sicheres Autentifikations Schema, Verschlüsseln von bewegten Agenten, Public-Key Verfahren
- Directory Service mit dem man den Ort eines Services herausfinden kann.
- Mehr Kommunikations Protokolle wie SMTP und HTTP
- Java Interpreter
- WWW Support.

# Literatur


[AraS] Vorläufiges Buchkapitel The Ara System for Mobile Agents, Holger Peine 1997

[RM1] Lara Reference Manual, Holger Peine 1995

[Vo1] Vortrag Konzept und Implementierung einer Plattform für Mobile Agenten, Holger Peine 1996