Infrastrukturen und Middleware

Seminar „Smart Environments“

ETH Zürich, 4. Mai 2004

von René Müller

Betreuer: Oliver Kasten

Inhalt

- Definition smarter Umgebungen
- Anwendungsbeispiel: iRoom
- Applikationsmodell für smarte Umgebungen
- Anforderungen an Middleware/Infrastruktur
- Realisierungen
  - iRoom
  - Gaia
- Vergleich, Fazit

Smarte Umgebungen

- Sind interaktiv; Benutzer beeinflusst Verhalten
- Man kann mit ihnen kommunizieren
- Besitzen IT (unsichtbar?)
- Passen sich dem Benutzer an
- Erlernen Gewohnheiten des Benutzers

[Junestrand, 2001]

Smarte Umgebungen

- Sind interaktiv; Benutzer beeinflusst Verhalten
- Man kann mit ihnen kommunizieren
- Besitzen IT (unsichtbar?)
- Passen sich dem Benutzer an
- Erlernen Gewohnheiten des Benutzers

[Junestrand, 2001]

Beispiele:

- Smarte Umgebungen im Cyberspace (virtuell)
- Smarte Umgebungen im Sinne von Active Spaces
  - Interaktive Museen
  - Smart Home
  - „Intelligente" Konferenz-Zimmer

Active Spaces

Active Space: A physical space coordinated by a responsive context-based software infrastructure that enhances the ability of mobile users to interact and configure their physical and digital environment.

nach [Román et al.], Entwickler von Gaia

Active Spaces

Active Space: A physical space coordinated by a responsive context-based software infrastructure that enhances the ability of mobile users to interact and configure their physical and digital environment.

nach [Román et al.], Entwickler von Gaia

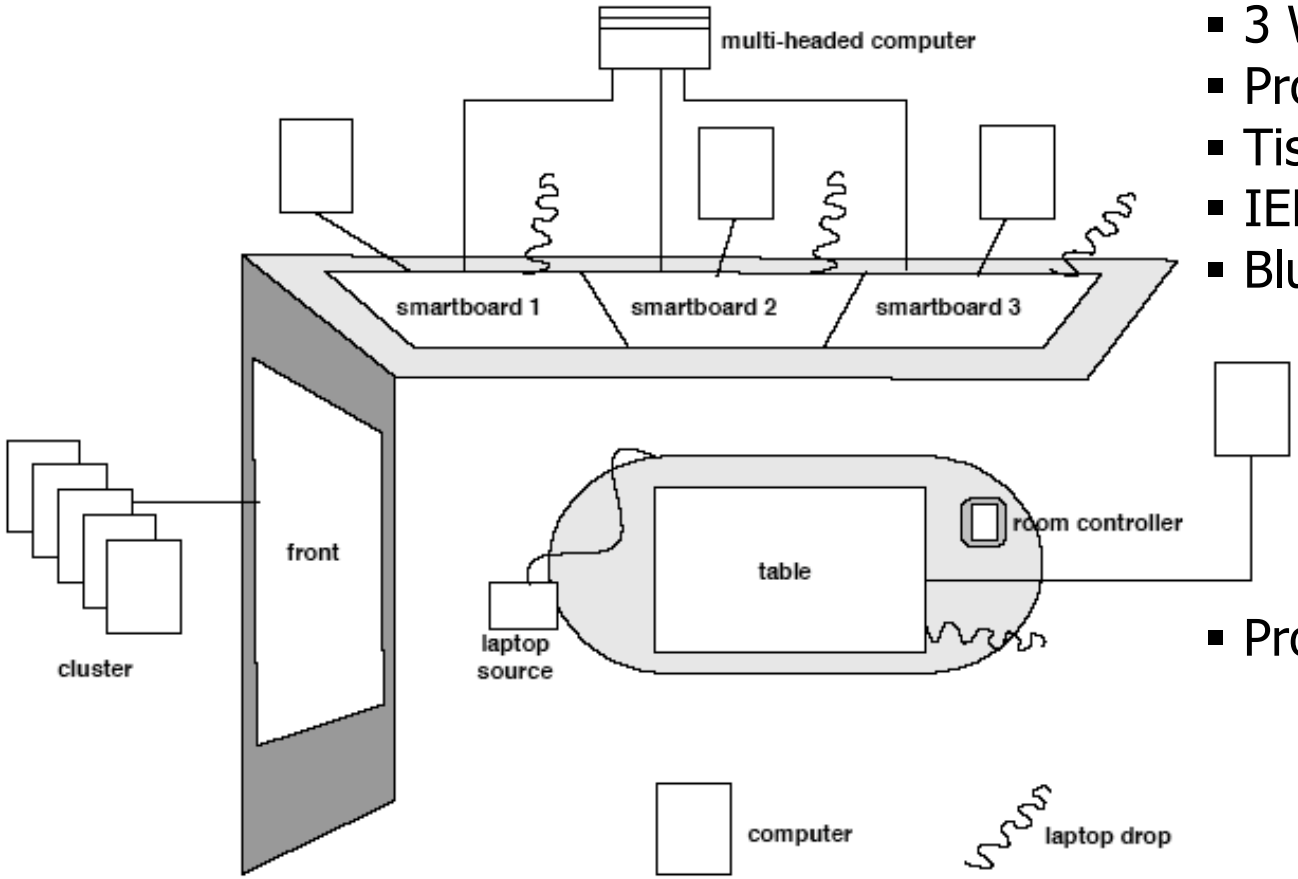
Wir beschränken uns im Folgenden auf interaktive Räume.

Eigenschaften:

- Adaption an verschiedene Geräte und Benutzer im Raum
- Erfasst und verwaltet Kontext-Informationen (Aktivität im Raum)
- Stellt Software-Infrastruktur für Koordination bereit
- Verschiedene Benutzer-Schnittstellen
- Ermöglicht Integration verschiedener Endgeräte (PDA, Scanner, ...)

Interaktiver Konferenzraum

Stanford Interactive Room („iRoom“)



Installation:

- 3 Wand-Touch-Screens
- Projiziertes Wand-Display
- Tischoberflächen-Anzeige
- IEEE 802.11b Ethernet (WLAN)
- Bluetooth Maus und Tastatur
- Proprietäre Hardware (iStuff)

Anwendungsszenario – Präsentation

- Benutzer aktiviert Raum
- Authentifizierung (Fingerprint)
- Raum erkennt
  - Ressourcen (PDA, Notebook, Digitalkamera)
  - Services (Drucker, File-Sharing unter den Teilnehmern)
  - Applikationen (Präsentationsprogramm, Terminkalender, Projekt-Planer)
- Vortragender startet Präsentations-Applikation (Notebook od. PDA)
- Smartboard-Anzeigen zuweisen
- Zuhörer erhalten Folien auf ihren Geräten und können Notizen anfügen.
- Zuhörer hat Zusatzinformation auf seinem Notebook → Schaltet diesen auf ein Display
- Koordination der Terminplaner der Anwesenden

Anforderungen an Applikationen

- Unterstützung mehrerer Ein- und Ausgabegeräte
- Datenaustausch mit anderen Anwendungen (z.B. Präsentation- mit Projektplanungs-Applikation)
- Anwendung muss kontext-sensitiv sein (Wer ist alles im Raum? Anzahl Personen?)
- Müssen mit minimalen Benutzereingriff adaptierbar sein

Anforderungen an Infrastrukutur

Middleware

Funktionale Aspekte

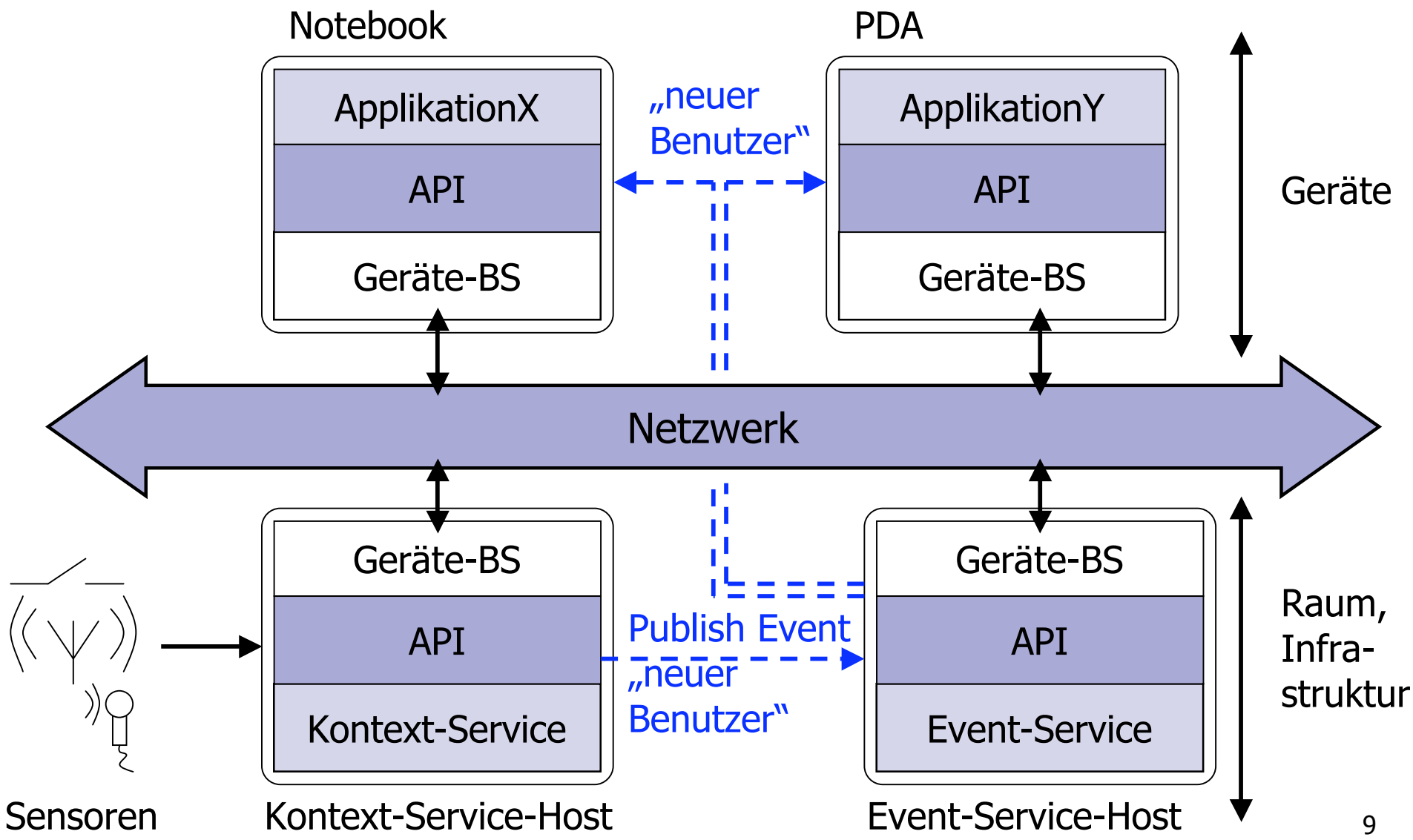
- Einheitliche Kommunikations-Schiene für Applikationen (Koordinationsinfrastruktur)
- Darstellung des aktuellen Kontexts
  - Sensoren erfassen Kontext
  - Kontext-Änderungen werden den Applikationen mitgeteilt
- Meta-Betriebssystem für beteiligte Geräte/Applikationen

Nichtfunktionale Aspekte

- Transparenz (unterschiedliche Geräte)
- Robustheit gegenüber Ausfall von Komponenten
- Flexibles API (Vorwärts-, Rückwärtskompatibilität)

Kontext hier = Menge der Geräte
∪ Menge der Benutzer
∪ Menge der Dienste
∪ Menge der Anwendungen

Applikations-Modell



Inhalt

- Definition smarter Umgebungen
- Anwendungsbeispiel: iRoom
- Applikationsmodell für smarte Umgebungen
- Anforderungen an Middleware/Infrastruktur
- Realisierungen
  - iRoom
  - Gaia
- Vergleich, Fazit

Interactive Room Operating System – Überblick

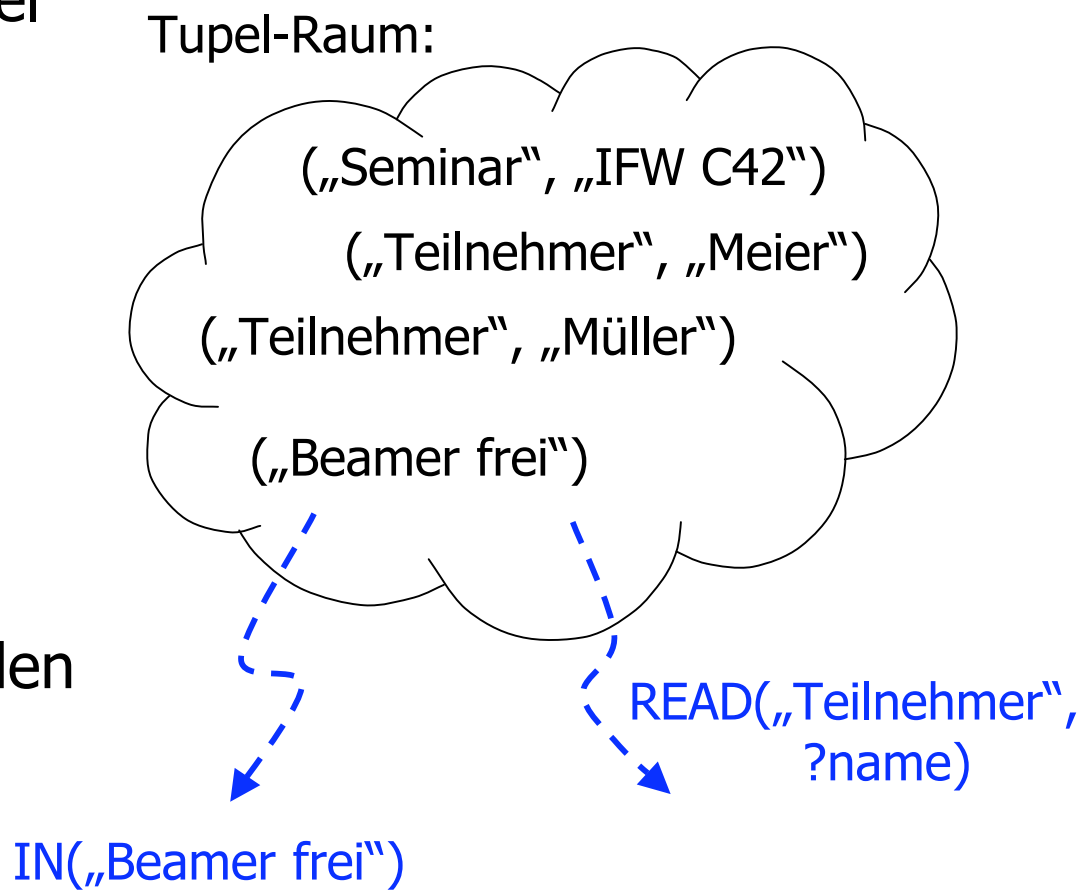
- iROS (interactive room operating system) für „Interactive Workspaces"-Projekt an der Stanford University.
- iROS = Meta-Betriebssystem verbindet Geräte, die mit iStuff-Schnittstellen ausgerüstet sind.
- iROS ermöglicht die Koordination der Applikationen
- iRoom stellt Koordinationsinfrastruktur zur Verfügung
- Verwendet erweitertes Tupelraum-Modell zur Koordination

Applikationskoordination

- „Coordination Language“ verbindet mehrere Aktivitäten (Prozesse, Benutzer) zu einem Ensemble.
- „Computation Language“ beschreibt Aktivität (erlaubt prinzipiell die Implementierung einer Turing-Maschine)
- [Gelernter, Carriero] „Coordination Languages“ und „Computation Languages“ sind orthogonal.
- Explizite Separierung ermöglicht
  - Portierbarkeit
  - Heterogenität
- Eine allgemeine Koordinationssprache ist
  - Ökonomisch
  - Flexibel
- Allgemeine Koordinationssprache auf Tupleraum-Modell (Linda)

Vom Linda Tupel-Raum...

- Daten werden durch Tupel **(attr1, ..., attrN)** repräsentiert
- Aktivitäten können
  - Tupel in Raum einfügen **OUT(attr1, ..., attrN)**
  - Tupel im Raum lesen **READ(attr1, ..., ?attrN)**
  - Tupel konsumierend lesen **IN(attr1, ..., ?attrN)**
- Tuple-Auswahl bei lesenden Zugriffen über Pattern-Matching



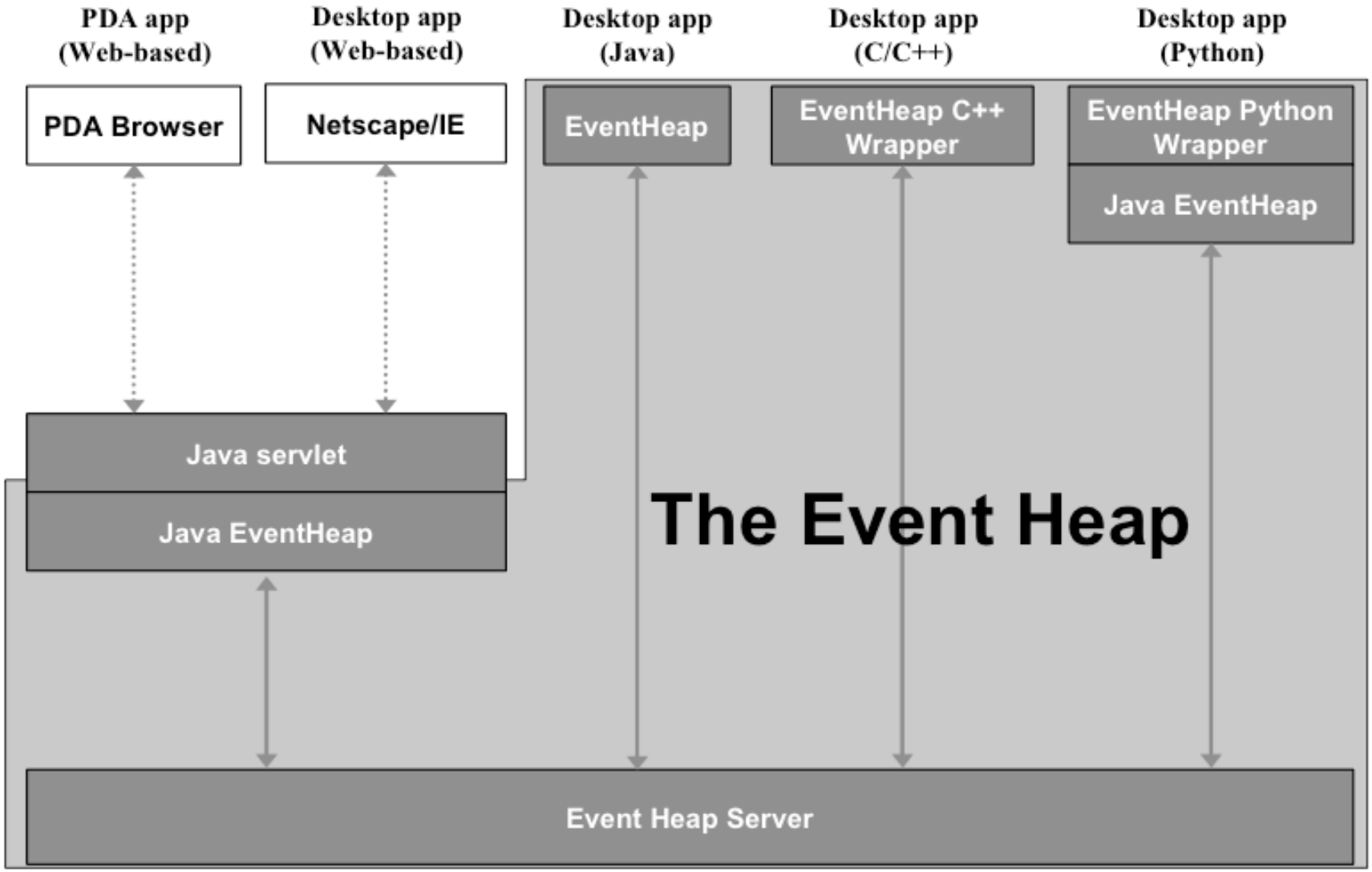
… zum Event-Heap

- Einschränkungen des Linda-Modells
  - Linda nicht-deterministisch wenn mehrere Tupel zu einem Suchmuster passen
  - Aktivität muss üblicher-weise mit IN/READ „pollen“ und kann während dem Nicht-Pollen somit ein Tupel verpassen.
- Erweiterungen
  - Tupelraum zu Event-Heap
  - Tupel wird Event(-tupel)
- Self-describing tuples
  - Felder haben Name + Wert
- Tuple Sequencing
  - IN/READ erhält ältestes passendes Tupel
- Expiration of Tuples
  - Time-to-Live Feld: Zeit, bis Garbage-Collection des Tupels
- Query Registration
  - Publish/Subscribe auf Such-muster

Implementation des Event-Heap

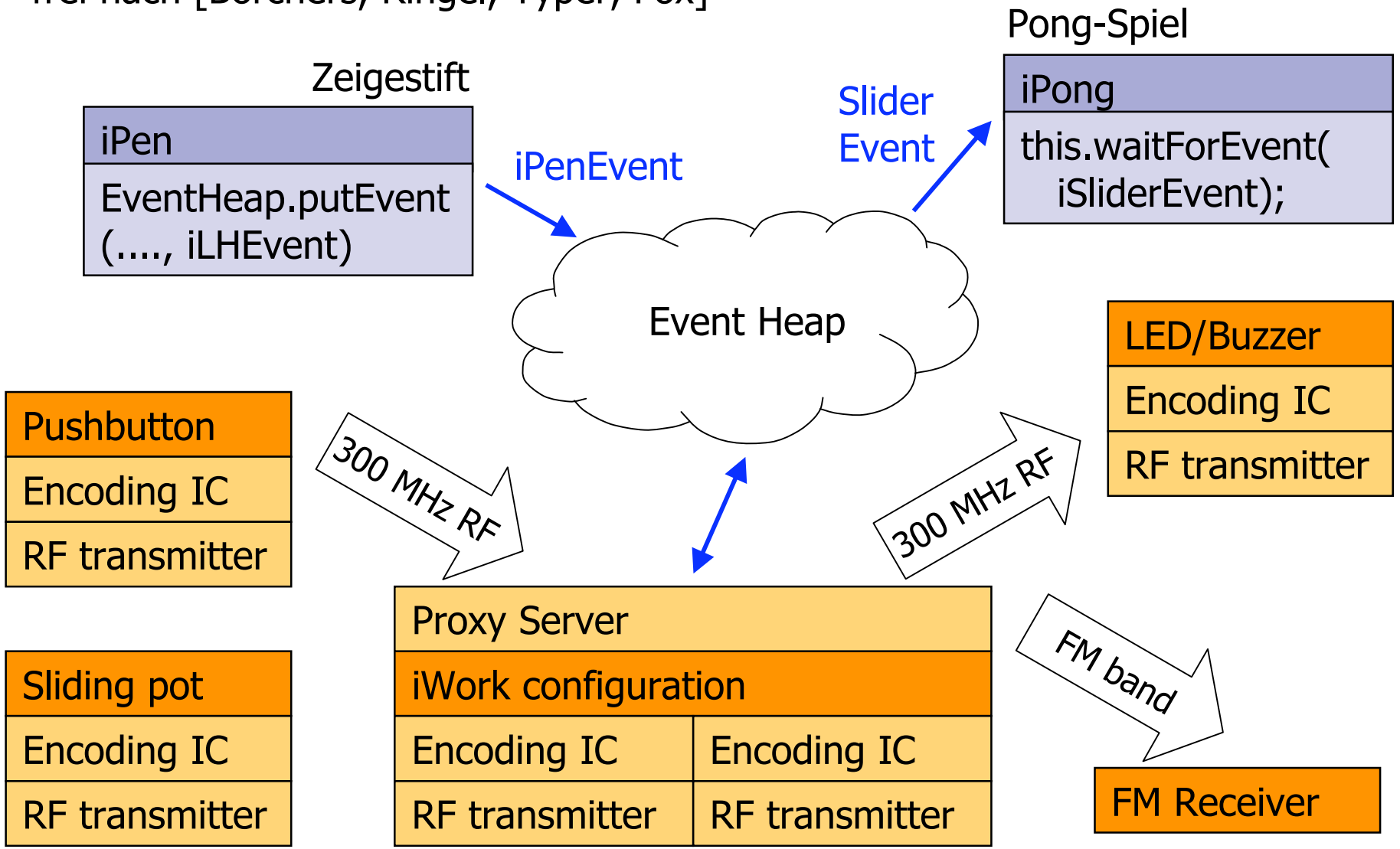
- Client/Server-Architektur
  - Tupelraum auf einzelnem (!) Server gespeichert
    - Vorteil: Single Point-of-Failure (???)
    - Neustart des Servers über dezidierten Web-Server
    - Clients versuchen automatisch Reconnect
  - Verbindung zu den Clients über permanente TCP-Verbindung
- Implementation
  - Server in Java
  - Client-Library in Java (45 kB) oder C++, Python-Java-Wrapper
- Mehrere Zugangsmöglichkeiten auf Event-Heap für Clientapplikationen möglich
  - Direkt via Event-Heap-Wiring-Protokoll auf TCP
  - Über HTTP-Formular für Palm-ähnliche Geräte

Zugang zum Event-Heap



Drahtlose UI-Komponenten

frei nach [Borchers, Ringel, Typer, Fox]



Auswertung iROS/iRoom

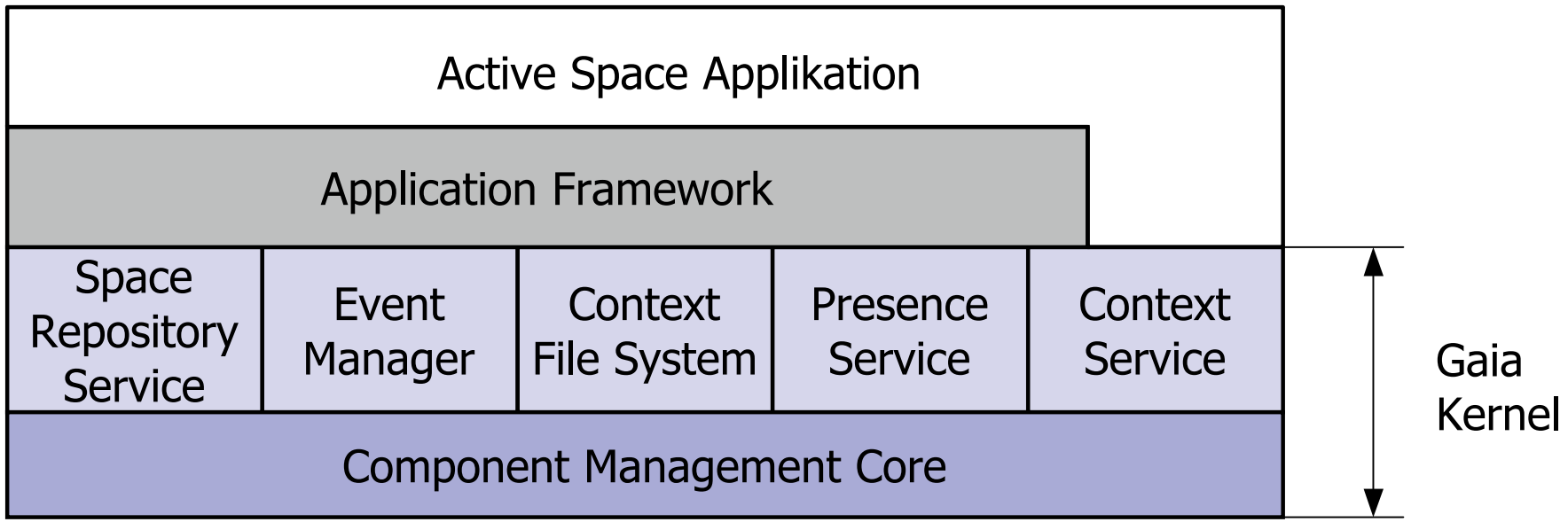
- Tupelraum nicht-repliziert auf einem Server?
- Anbindung der Clients über TCP-Verbindung – warum nicht UDP aufgrund der Event-Basis?
- Event-Orientierung nicht für jede Art der Kommunikation geeignet
  - Ungeeignet für Streams (z.B. Audio über externen UKW-Kanal der per Event ein-/ausgeschaltet wird)
- Semantik: Events mit Verfallsdatum? (und dennoch Publish/Subscribe-Prinzip mit Query-Registration)
- Warum überhaupt (Event-)Tupel wenn man trotzdem Publish/Subscribe Mechanismus einführt?

Gaia – Middleware Plattform

- Gaia: (griech.) Göttin von Allem, Erdengöttin
- Middleware-Plattform für „Active Spaces“
- Entwickelt an der University of Illinois, Urbana
- Sammlung von CORBA-basierten Kernel-Services
- Gaia-Komponenten sind verteilte Objekte
- Kernel-Services
  - Space Repository Service
  - Event Manager
  - Context File System
  - Context Service
- Framework (API) für „Active Space“-Applikationen

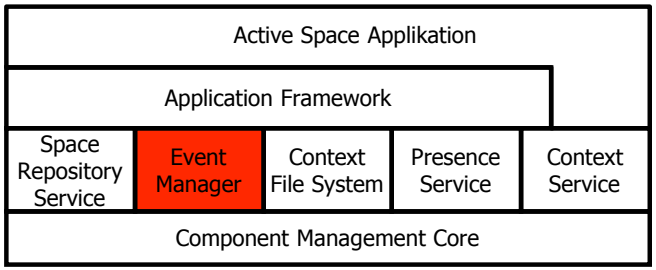
Gaia – Architektur

Nach [Román et al.]



- Gaia-Services sind als Komponenten implementiert
- Component Mgmnt. Core erlaubt als „Komponenten-Schiene“
- Komponenten können dynamisch geladen, entfernt werden
- Komponenten können auch von beliebigen Geräten hinzugefügt werden

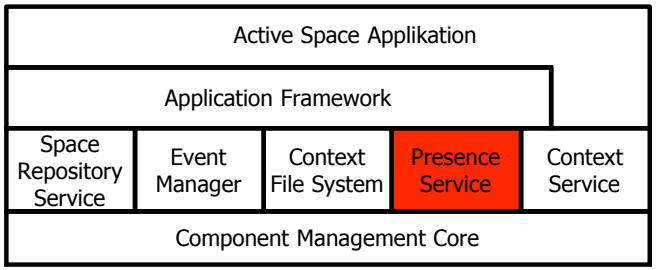
Event-Manager



- Broadcast-Mechanismus für Zustandsänderungen
- Stellt Kommunikations-Modell bereit
  - Suppliers
  - Consumers
  - Channels

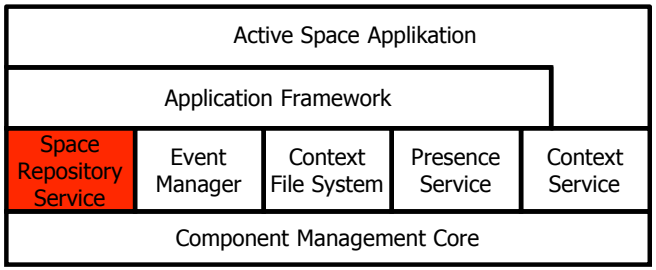
- Prinzip: Entkopplung von Event-Quelle und -Senke
- Event-Manager baut auf CORBA-Event-Service auf
- Kein Tupelraum, kein Aufheben von Events

Presence-Service



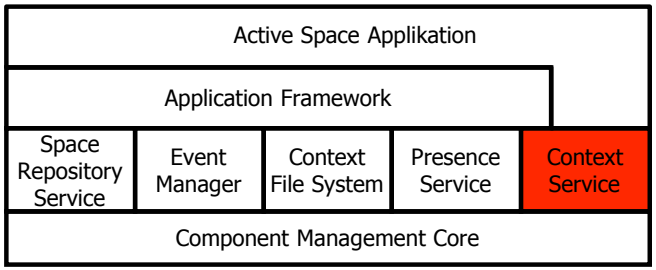
- Detektiert digitale und physikalische Entitäten im „Active Space“. Das sind
  - Applikationen
  - Services
  - Geräte
  - Personen
- Beaconing-Mechanismus basiert auf Leases für digitale Entitäten
- Sensoren für physikalische Entitäten (u.U. auch Position im Raum)

Space-Repository



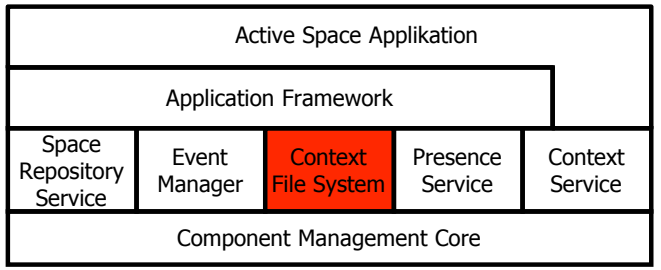
- Speichert Informationen über alle Entitäten (z.B. Name, Type, Besitzer)
- Erlaubt den Clients den „Active Space" nach bestimmten Attributen zu durchsuchen
- Space-Repository wird vom Presence-Service über ein- und austretende Entitäten informiert d.h. subskribiert Channels des Presence-Service
- Basiert auf CORBA Trader

Context-Service



- „Active Spaces" sind „context-aware"
- Kontext bestimmt Verhalten der Applikationen
- Applikationen greifen über Context-Service auf aktuellen Kontext zu
- Kontext-Modell basiert auf Logik aus Quadrupel-Predikaten
  - Klauseln **Context(<Context-Type>, <Subject>, <Relater>, <Object>)**
    - **Context(location, René, entering, IFW C42)**
    - **Context(Number of people, IFW C42, >, 4) AND Context(Application, PowerPoint, is, running) => Context(Social Activity, IFW C42, is, Presentation)**
- Im Context-Service sind mehrere Context-Provider registriert
  - Low-level Kontext: Sensoren z.B. Temperatur-, Lautstärke-Sensor
  - High-level Kontext: Logik-Komponenten mit Regelwerk

Context-File-System



- „context-awares" File-System
  - Persönliche Daten werden der Applikation verfügbar gemacht sobald der Benutzer anwesend ist
  - Daten werden in einem Format aufbereitet, das den Vorlieben des Benutzers oder Fähigkeiten der Geräte entspricht
- Virtuelle Verzeichnis-Hierarchie aufgrund Kontext-Typen
  - Verzeichnis = Kontext
  - Pfad = Stellt Kontext-Typen und -Werte dar
  - Datei = Daten welche diesem Kontext-Zugeordnet sind
  - Bsp: Welche Dateien haben Kontext **location==IFW C42** und **situation==Seminar**?
    → Alle Dateien im Verzeichnis
    /location:/IFW C42/situation:/Seminar

Auswertung Gaia

- Implementation des Component Management Core in CORBA (andere Technologien wie SOAP, RMI sollen ebenfalls verwendet werden können)
- Middleware verwaltet Kontext-Information
- Applikationen können Kontext-Daten abfragen
- Komplexere Kontexte können durch Regelwerk abgeleitet werden
  - Wie werden diese Regeln sinnvoll angelegt?
  - Haben diese Regel wirklich universelle Gültigkeit?
  - Wenn nicht, wer definiert diese Regeln?

Vergleich iROS – Gaia

iROS

- Fokus: Kommunikations-Mechanismus, d.h. Event-Heap
- Event-Heap Modell
- Low-Level Ansatz: Alles ist ein Event. Kommunikation erfolgt über Event-Tupel in Tupelraum. Applikationen müssen Raum überwachen. Applikationen müssen individuell den Tupel Bedeutung beimessen (Anhand von Tupel-Name)

Gaia

- Fokus: Kernel-Services sowie deren Interaktion
- Shared Object-Model
- Higher-Level Ansatz: Gaia behält Kontext-Informationen. Diese sind für alle Applikationen gültig (z.B. Applikation muss nicht selber entscheiden, ob **Kontext == Meeting**).

Fazit

- Middleware für „Smart Environments“ verlangen andere Konzepte verlangt als konventionelle Middleware
- Zwei-Realisierungen vorgestellt
  - Realisierung über Ereignis-basierte Kommunikation
  - Realisierung auf bestehender CORBA-Technologie mit spezifischen Diensten für End-Applikationen
- Ansätze sind vorhanden, aber...
  - Unausgereiftheiten bei der technischen Implementierung
  - Eventuelle Schwierigkeiten bei der Umsetzung, resp. Festlegung des Regelwerks Kontext-Information

Referenzen

- S. Junestrand: *Private and Public Digital Spaces*. International Journal of Human Computer Interaction; Special issue of Home use of IT, 2001 (http://www.arch.kth.se/~junestrand/papers/ijhci00.pdf)
- J. Borchers et al.: *Stanford Interactive Workspaces: A Framework for Physical and Graphics User Interface Prototyping*. *IEEE Wireless Communications*, special issue on Smart Homes, 2002 (http://http//www.stanford.edu/~borchers/publications/ieee-smarthomes2002/BORCHERS_LAYOUT.pdf)
- B. Johanson, A. Fox: *The Event Heap: A Coordinaten Infrastructure for Interactive Workspaces*. Proc. 4th IEEE Workshop on Mobile Computing Systems and Applications, 2002 (http://graphics.stanford.edu/papers/eheap3/eheap_wmcsa.pdf)
- D. Gelernter, N. Carriero: *Coordination Languages and their Significance*. Communications of the ACM, Vol. 35 No. 2, pp. 96-105, 1992
- M. Román et al: *Gaia: A Middleware Infrastructure to Enable Active Spaces*. IEEE pervasive computing, Vol. 1 No. 4, pp. 74-83, 2002 (http://www.cs.uiuc.edu/Dienst/UI/2.0/Describe/ncstrl.uiuc_cs/UIUCDCS-R-2002-2265)