



# An Architecture for Context Prediction

2. Mai 2005, Mainz

**Vortrag zum Dissertationspreis der  
Gesellschaft für Informatik**

Rene Mayrhofer

Institut für Pervasive Computing

Johannes Kepler Universität Linz, Austria

[rene@soft.uni-linz.ac.at](mailto:rene@soft.uni-linz.ac.at)

# Problemfeld

Einleitung

Was ist *Pervasive Computing*?

Ansatz

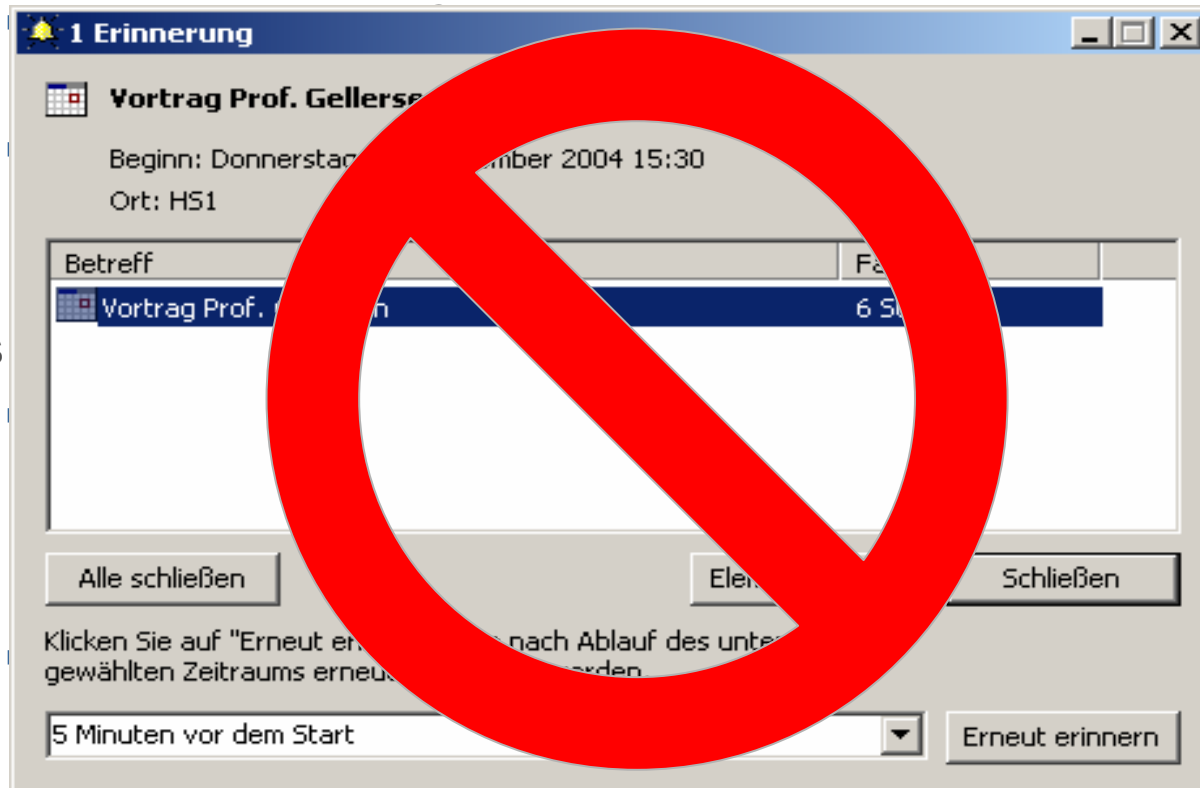
Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag

Was



ändern

t

Dabei

ation

Was ist *Context Awareness*?

# Context Awareness

## Einleitung

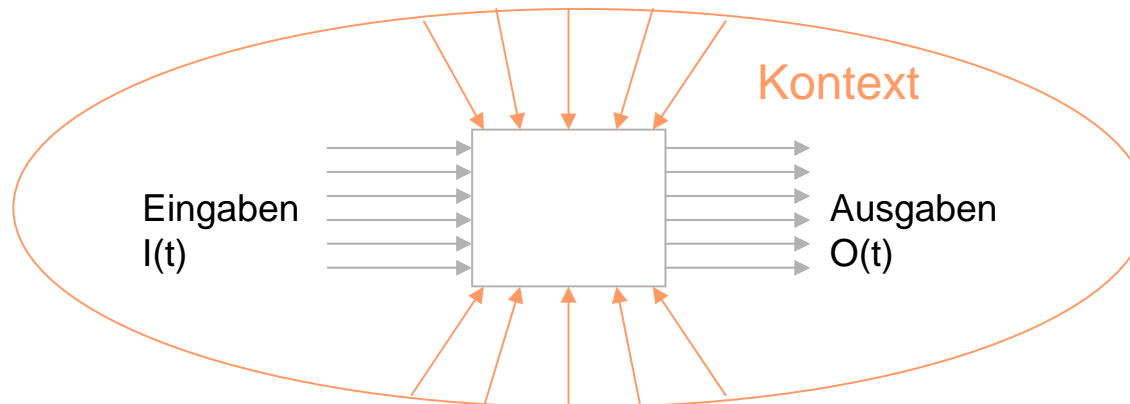
Viele Definitionen für Kontext in diesem Sinne, z.B.:

- *“Three important aspects of context are: where your are, who you are with, and what resources are nearby [...]. Context encompasses more than just the user’s location, because other things of interest are also mobile and changing.”* [SAW 1994]
- *“any information that can be used to characterize the situation of an entity, where an entity can be a person, place or a physical or computational object”* [Dey 1999]
- A working model for context [Schmidt 2002]

## Ansatz

## Architektur

Kontext ist alles **außer** den expliziten Ein- und Ausgaben [LS 2000].



## Implementierung

## Ergebnisse

## Wissenschaftlicher Beitrag

### Problembeschreibung:

Wie kann ein Gerät seinen (oder seines Benutzers) **aktuellen Kontext** ableiten und **zukünftigen Kontext** ausgehend von aufgezeichneten Kontexthistorien vorhersagen?

[SAW 1994] B.N. Schilit, N. Adams, R. Want: „Context-aware computing applications“

[Schmidt 2002] A. Schmidt: “Ubiquitous Computing – Computing in Context”, PhD Thesis, Lancaster Univ.

[Dey 1999] A. Dey, G.D. Abowd, D.Salber: „A Context-based infrastructure for smart environments“

[LS 2000] H. Lieberman, T. Selker: „Out of context: Computer systems that adapt to, and learn from, context“

## Context Awareness (2)

### Einleitung

Kontext hat verschiedene Aspekte, z.B.:

- Zeit
- Ort
- physikalisch (Temperatur, Luftfeuchtigkeit, etc.)
- sozial (unter Kollegen / Familienmitgliedern etc.)

### Ansatz

### Architektur

⇒ Daher vernünftig, mehrere einfache Sensoren zur Erfassung dieser Aspekte zu verwenden (siehe Gellersen et.al.)

### Implementierung

### Ergebnisse

### Wissenschaftlicher Beitrag



# Proaktivität

## Einleitung

Proaktives vs. reaktives Verhalten eines (Computer-) Systems

## Ansatz

- Bestimmt, ob das System lediglich auf Änderungen in seiner Umgebung reagiert, oder ob es in Voraussicht handeln kann
- Wichtige Eigenschaft von **Software-Agenten**
- Definition von Proaktivität basierend auf Systemzuständen [MRF 2003a]

## Architektur

- Der interne Zustand einer (Moore) Zustandsmaschine ist abhängig vom letzten Zustand und den aktuellen Eingaben:  $q_t = \delta(q_{t-1}, a_{t-1})$

## Implementierung

- **Reaktives System:** Ausgabe anhängig vom aktuellen Zustand

$$b_t = \lambda(q_t)$$

## Ergebnisse

- **Proaktives System:** Ausgabe zusätzlich von vorhergesagten zukünftigen Zuständen abgängig

$$b_t = \lambda\left\langle q_t, \bar{q}_{t+1}, \bar{q}_{t+2}, \dots, \bar{q}_{t+m} \right\rangle$$

## Wissenschaftlicher Beitrag

# Verwandte Arbeiten

## Einleitung

- TEA

## Ansatz

- Smart-Its
- Universität Helsinki
- Context Toolkit

## Architektur

- Robotics
- CIS

## Implementierung

- Neural Network House
- Aware Home
- MavHome

## Ergebnisse

- ContextCube
- Portolano

## Wissenschaftlicher Beitrag

- ...

# Ansatz zur Kontextvorhersage

Einleitung

Ansatz

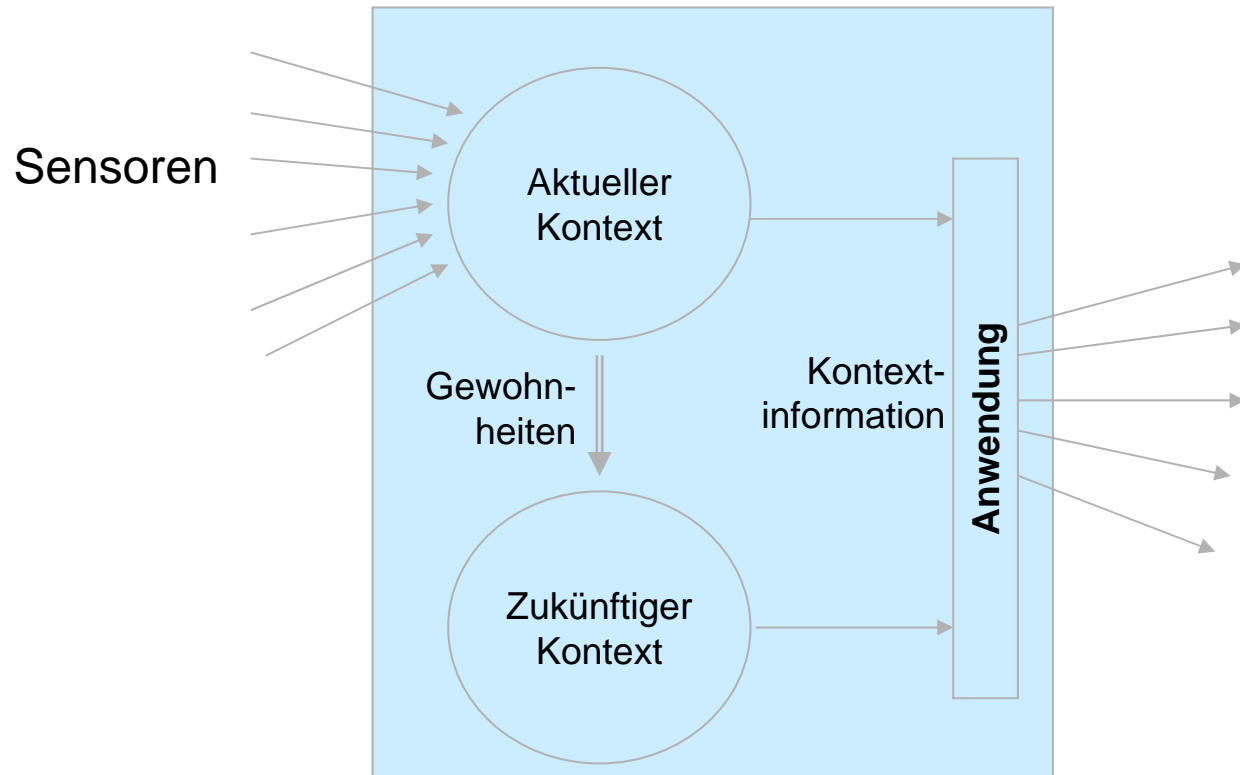
Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag

- Erkennen des aktuellen Kontext aus mehreren Sensoren (z.B. [Schmidt 2002])
- Vorhersage zukünftiger Kontexte durch Lernen von Benutzerverhalten
- Möglichst **ohne** Verwendung von Domänen-spezifischem Wissen



# Konzept

Einleitung

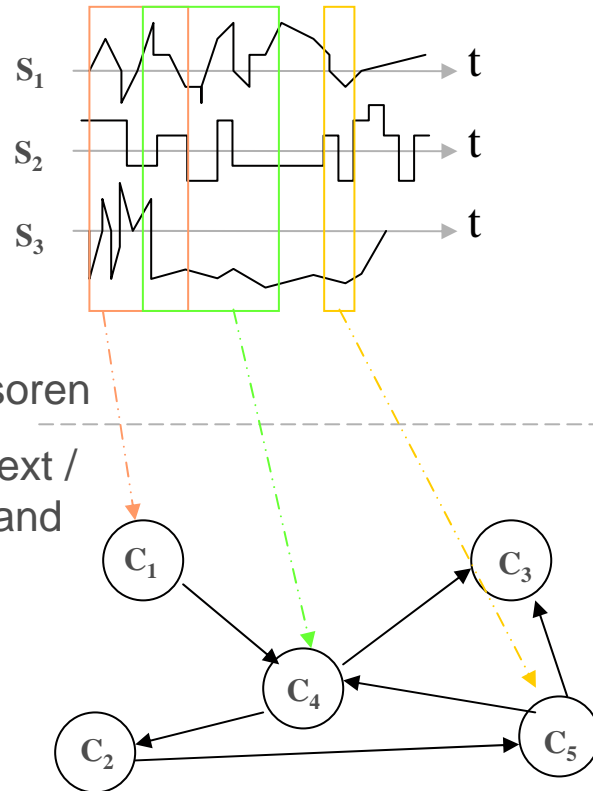
Ansatz

Architektur

Implementierung

Ergebnisse

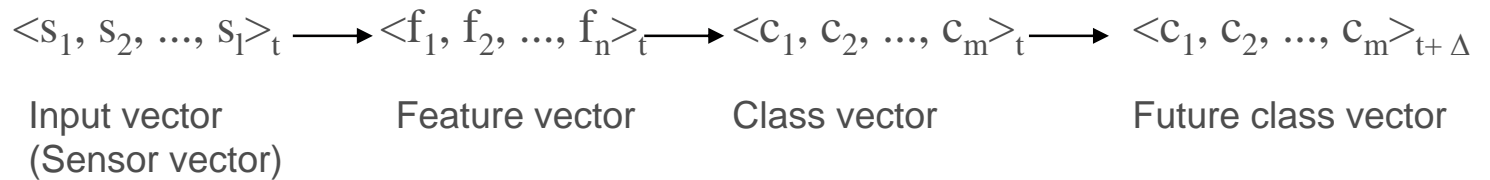
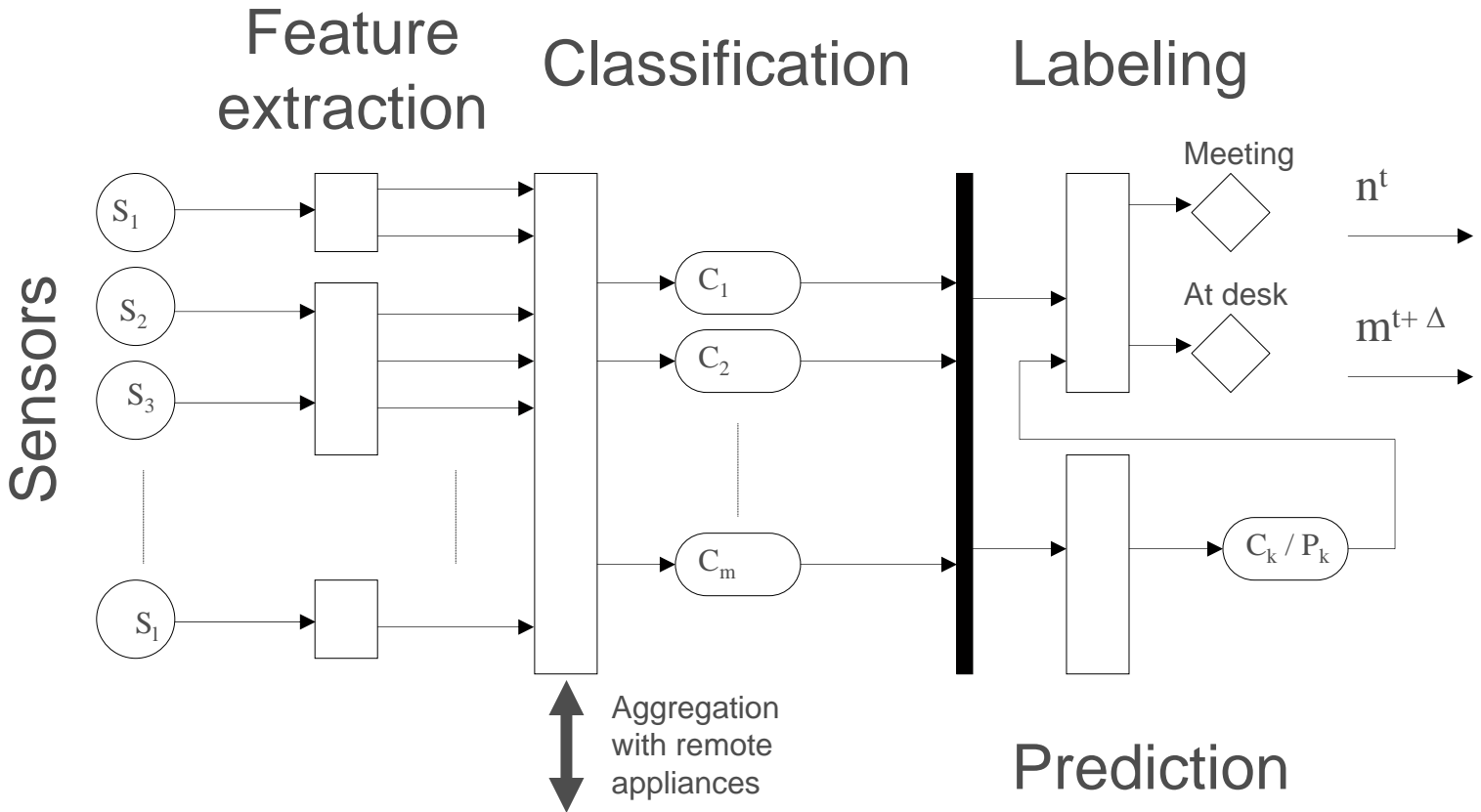
Wissenschaftlicher  
Beitrag



- Sensoren liefern Zeitreihen
- Spezielle Muster in Eingabeströmen können als Zustände einer (teilweise beobachtbaren, aber nicht steuerbaren) Zustandsmaschine interpretiert werden
- Zustände werden als Benutzer- bzw. Gerät-Kontexte aufgefasst
- Vorhersage der Zustandstrajektorie durch Extrapolation der Historie in die Zukunft



# Architektur



# Schritt 1: *Sensors*

Einleitung

Ansatz

Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag

- Sensoren liefern **Zeitreihen**
- Entweder zu fixen Zeitpunkten abgetastet oder Ereignisse
- Beispiele für aktuell verfügbare „Sensoren“ zur Erfassung des aktuellen Benutzerkontext:
  - Zeit
  - Anwendung
  - Helligkeit
  - Mikrofon
  - Bluetooth
  - WLAN
  - In Dockingstation
- **Zusätzliche Sensoren können abgefragt werden:**
  - GPS
  - GSM
  - Kompass
  - Beschleunigungssensoren
  - Neigungssensoren
  - Temperatursensoren
  - Drucksensoren



- **Weitergabe von Sensordaten an andere Geräte möglich**

## Schritt 2: *Feature Extraction*

Einleitung

- Wandelt rohe Sensordaten in aussagekräftigere Werte (*Merkmale*)
- Verwendet Wissen über die Anwendungsdomäne
- Aus einem Sensordatenstrom können mehrere Features gewonnen werden

Ansatz

⇒ **Hochdimensionaler Feature-Raum**

Architektur

- Unterschiedliche Typen von Features:
  - Numerische (kontinuierliche): z.B. Helligkeitssensor, Pulssensor
  - Numerische (diskrete): z.B. Anzahl von Access Points in Reichweite
  - Ordinale: z.B. Wochentag
  - Nominale: z.B. aktuelle WLAN-SSID im Infrastruktur-Modus, Liste von über WLAN/BT erreichbaren Geräten in unmittelbarer Umgebung

Implementierung

Ergebnisse

- Aber: nur 2 Operationen für jede Feature-Dimension benötigt
  - **Distanzmetrik**
  - **Adaptionsoperator**

Wissenschaftlicher  
Beitrag

## Schritt 3: *Classification*

Einleitung

- Klassifiziert Features und findet häufig vorkommende Muster (sog. Cluster) in den Eingabedaten ⇒ ohne Benutzerinteraktion möglich, **unaufdringlich**

Ansatz

- Verschiedene Arten von Klassifikationsalgorithmen
  - Typ (**partitionierend** / hierarchisch)
  - „**Weiche**“ / „harte“ Klassifikation
  - Überwacht / **unüberwacht**

Architektur

- Voraussetzungen für geeignete Algorithmen zur Kontexterkenkung:
  - Online bzw. inkrementell
  - Adaptiv
  - Dynamische Anzahl von Klassen und dynamische Struktur
  - Finden von Clustern in Unterräumen
  - „Weiche“ Klassifikation
  - Robustheit gegen Rauschen
  - Geringer Ressourcenbedarf
  - Einfachheit
  - Interpretierbarkeit der Klassen / Wahrung des Datenschutzes

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag

# Klassifikationsalgorithmen

Einleitung

Ansatz

Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag

| Algorithmus                      | Topologie /<br>Anzahl<br>Klassen | Online /<br>inkrementell              | Adaptiv | Hart /<br>Weich |
|----------------------------------|----------------------------------|---------------------------------------|---------|-----------------|
| K-Means                          | fix                              | ja                                    | ja      | hart            |
| FCM                              | fix                              | ja                                    | nein    | weich           |
| Neural Gas                       | fix                              | ja                                    | nein    | teilweise       |
| SOM                              | fix                              | nein                                  | nein    | weich           |
| ART                              | variabel                         | ja                                    | nein    | weich           |
| IDBSCAN                          | variabel                         | inkrementell,<br>aber nicht<br>online | nein    | weich           |
| SNN                              | fix oder<br>variabel             | ja                                    | nein    | weich           |
| Growing Neural Gas<br>[Fri 2003] | variabel                         | ja                                    | ja      | weich           |

## Schritt 4: *Labeling*

Einleitung

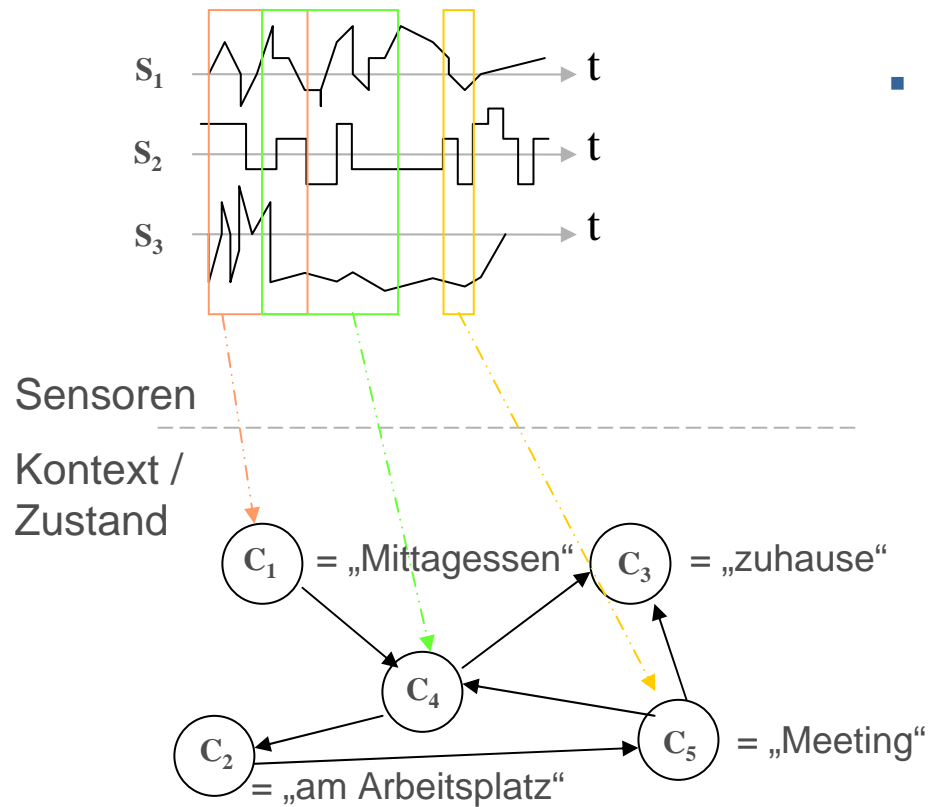
Ansatz

Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag



- 1: {0,1} Zuordnung von (Meta-) Clustern zu Kontext-Namen
- Derzeit 2 Möglichkeiten:
  - Bei stabilen (Meta-) Clustern von Schritt 1: direkte Zuordnung (Meta-) Cluster-ID zu Name
  - Bei durch Adaption/Lernen zu stark veränderlichen (Meta-) Clustern: zusätzlicher, sehr einfacher Clustering-Schritt über Clusterdaten [Lae 2001]

## Schritt 5: *Prediction*

Einleitung

- Erkannte Kontexte können als „Zustände“ einer Zustandsmaschine verstanden werden

Ansatz

- Beobachtung der entstehenden Zustandstrajektorie erlaubt Vorhersage
- Aspekte von Zeitreihenvorhersage:

Architektur

- Periodische Muster (z.B. Wochenenden, Meetings)
- Sequentielle Muster (z.B. Reisevorbereitungen, Zubereitung von Speisen)
- Trends (sich ändernde Gewohnheiten)

Implementierung

- Voraussetzungen für geeignete Algorithmen zur Kontextvorhersage:

Ergebnisse

- Unüberwachte Modellbildung
- Online
- Inkrementelle Modellerweiterung
- Konfidenzabschätzung
- Automatisches (implizites) Feedback
- Manuelles (explizites) Feedback
- Langzeit- vs. Kurzzeitvorhersage

Wissenschaftlicher  
Beitrag

# Möglichkeiten zur Kontextvorhersage

Einleitung

- Basierend auf Trajektorie von Kontexten, d.h. Kontext-Klassen
- Vorteil gegenüber unabhängiger Vorhersage von Feature-Dimensionen: gemeinsame Betrachtung aller Aspekte von Kontext

Ansatz

- Prinzipiell 2 Optionen:

Architektur

- Jede Dimension des Kontext-Klassenvektors getrennt als

**kontinuierliche Zeitreihe**

⇒ Abhängigkeiten zwischen verschiedenen Kontexten nicht berücksichtigt, aber erlaubt beliebige Überlappungen

- Aggregation aller Kontexte in eine einzelne, **kategorische Zeitreihe**  
AAACCBCCAAADDDDDDEEEEEEBBAAAA.....

⇒ berücksichtigt explizit die Zusammenhänge zwischen verschiedenen Kontexten, da diese als sich gegenseitig ausschließend betrachtet werden

Implementierung

Ergebnisse

⇒ **Flaches vs. hierarchisches/überlappendes Kontextmodell**

- Viele bekannte Methoden zur Zeitreihenvorhersage, aus denen für spezielle Anwendungen ausgewählt werden kann

Wissenschaftlicher  
Beitrag



# Implementierung als Software Framework

Einleitung

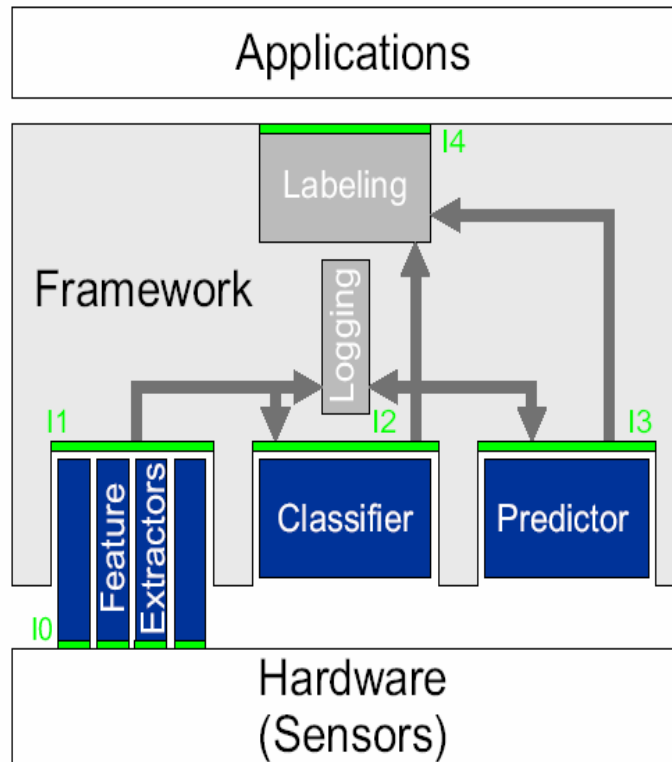
Ansatz

Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag



Cross-Plattform:

- Derzeit für Win32, Windows CE (>=3.0), Linux IA32 und ARM sowie Symbian OS
- Basierend auf dynamisch ladbaren Modulen:
  - Feature Extraktoren (= Sensordatenerfassung + Feature Extraktion)
  - Klassifikationsmethoden
  - Vorhersagemethoden
- Benennung von Kontexten über externe Applikation über netzwerktransparente Protokolle ⇒ Trennung von HCI und automatischer Kontexterkenkung
- Speziell im Hinblick auf Geräte mit geringen Ressourcen entwickelt

# Evaluierung mit Realdaten

Einleitung

- Aufgezeichnete Echt Daten
- Aufzeichnung vollständig im Hintergrund  $\Rightarrow$  Unaufdringlichkeit gewahrt
- Zeitraum: ca. 2 Monate
- 28 Dimensionen im Feature-Raum
- $\sim$  90000 Datenpunkte

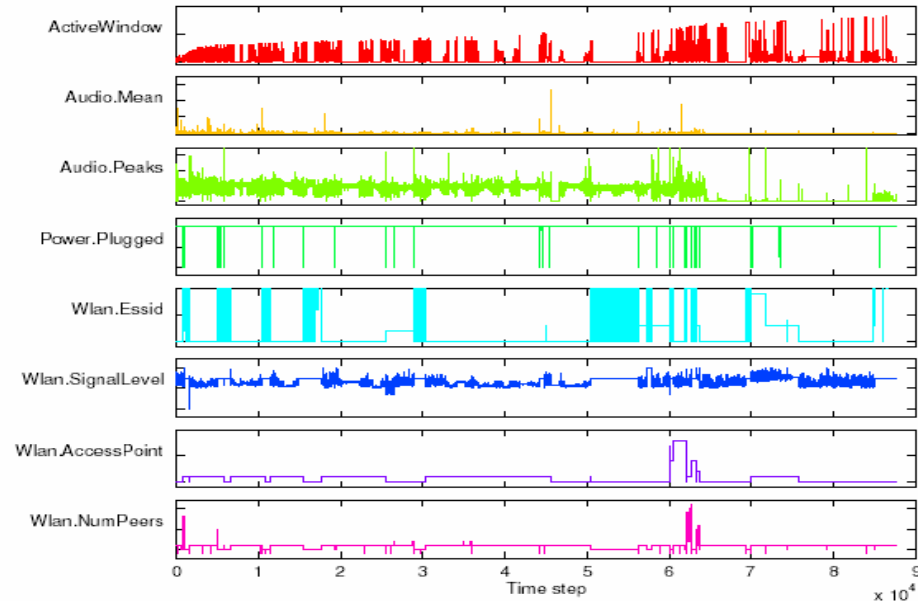
Ansatz

Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag



## Klassifikation:

- Evaluierung von K-Means, SOM und LLGNG
  - K-Means: kleinster Fehler mit 6 Clustern: **0,7451**
  - SOM: kleinster Fehler mit 71x21 (=1491) Neuronen in Ausgabeschicht: **0,5659**
  - Erweiterter LLGNG: 9 Meta-Cluster aus 109 Clustern: **0,0069**
- LLGNG erzielt kleineren Fehler mit weniger Clustern als SOM
- Erweiterte Variante bietet zusätzlich Konzept der Meta-Cluster

## Evaluierung mit Realdaten (2)

Einleitung

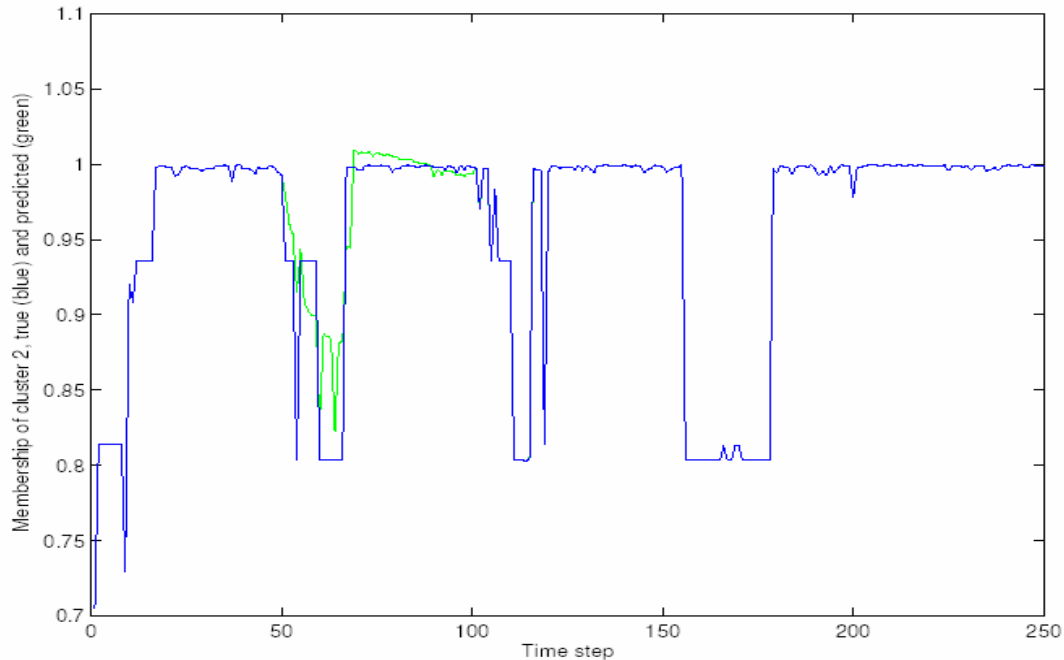
Ansatz

Architektur

Implementierung

Ergebnisse

Wissenschaftlicher  
Beitrag



| Algorithmus      | Fehler |
|------------------|--------|
| ARMA             | 0,04   |
| MLP              | 0,62   |
| SVR              | 0,24   |
| Central tendency | 0,46   |
| ALZ              | 0,46   |
| ALZ + Duration   | 0,44   |
| HMM              | 0,46   |
| SVM              | 0,46   |

### Vorhersage:

- Derzeit beste Ergebnisse mit ARMA und saisonaler Korrektur
- HMM, SVM etc. mit diesen Daten weniger geeignet

# Wissenschaftlicher Beitrag

Einleitung

Entwicklung einer **Architektur zur Kontextvorhersage**:

- Vorhersage von Kontexten auf hoher Ebene anstelle einzelner Aspekte durch Interpretation von Kontexten als Zustände und Beobachtung der daraus entstehenden Trajektorie
- Flexible Einsatzmöglichkeiten durch möglichst einfache, klar definierte Schnittstellen zwischen Einzelschritten
- Klassifikation, Benennung und Vorhersage explizit als eigenständige, austauschbare Teile
- Umgang mit heterogenen Eingabedaten

Ansatz

Architektur

Implementierung

Kriterien / Methoden zur Klassifikation und Vorhersage

Ergebnisse

Implementierung der Architektur in Form eines offenen Software-Frameworks:

- Direkte Verwendung verschiedenster Sensortypen ohne interne Kodierung von nicht-numerischen Sensordaten
- Erweiterung von LLGNG um das Konzept der Meta-Cluster sowie Optimierungen zur Verbesserung der Laufzeit
- Unterstützung verschiedener Plattformen und Entwicklung im Hinblick auf Geräte mit geringen Ressourcen

Wissenschaftlicher  
Beitrag



*“It is hard to predict, especially the future.”*

Niels Bohr

Winner of the 1922 Nobel Prize in Physics



***“If we knew what it was we were doing, it would not be called research, would it?”***

Albert Einstein

Winner of the 1921 Nobel Prize in Physics



***Danke für Ihre Aufmerksamkeit!***