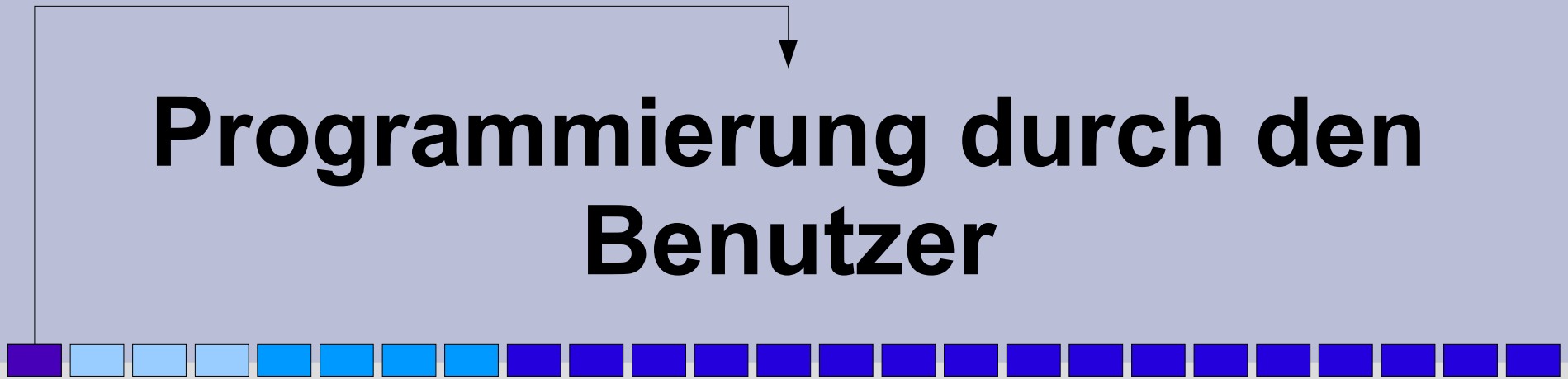
Programmierung durch den Benutzer



Programming by Example

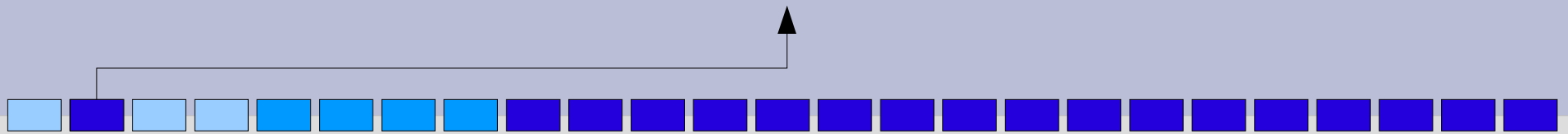
Lukas Stucki

Tangible Programming Interfaces

Dejan Pilav

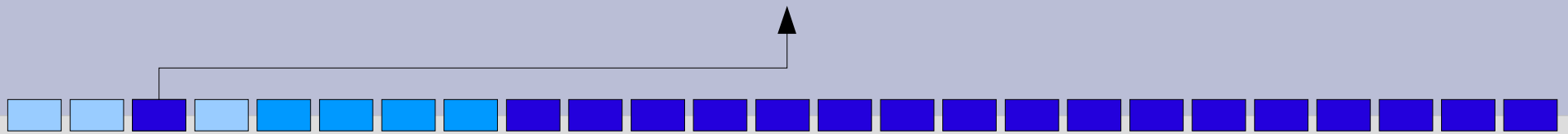
Betreuung: Marc Langheinrich

Übersicht



- Übersicht: Programmierung durch Benutzer
  - Wieso soll der Benutzer programmieren?
  - mögliche Ansätze
- Programming by Example
  - Beschreibung
  - „klassische“ Beispiele
- „a CAPpella“
  - Beschreibung
  - Beispiel

Programmierung duch Benutzer?

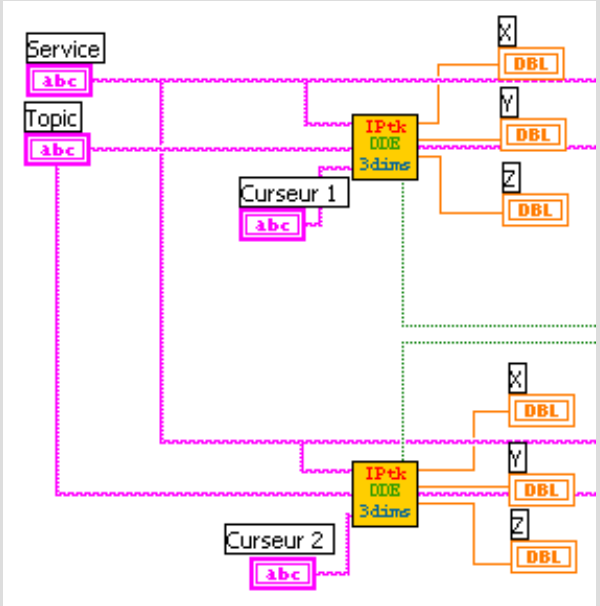


- Bedürfnisse sind einzigartig
- Benutzer kann das „Programm“ selber an Änderungen anpassen
- Es ist schwierig, durch Dritte ein „persönliches“ System ohne gute Kenntnis über deren Benutzung und Bedürfnisse zu erstellen

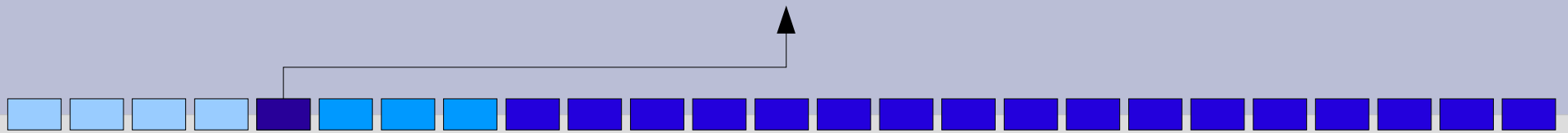
Mögliche Ansätze

- Programming by Example
  - Lukas Stucki
- Tangible Programming Interfaces
  - Dejan Pilav

- Visual Programming
- Recording Macros
- Natural Programming
- Application-specific Languages
- Script Languages etc.



Programming by Example (PBE)



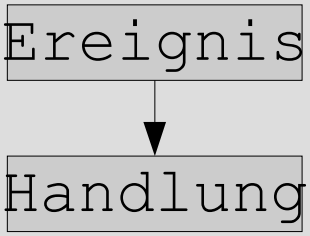
- Auch „Programming by Demonstration“
- Benutzer steht im Zentrum
- verallgemeinert Handlungen des Benutzers
- Kann Handlungen wiederholen
- Kann die programmierten Befehle bei Erkennung der Situation starten
- Gegensatz zu „Programming by Description“

PBE: Beispiele (1/2)

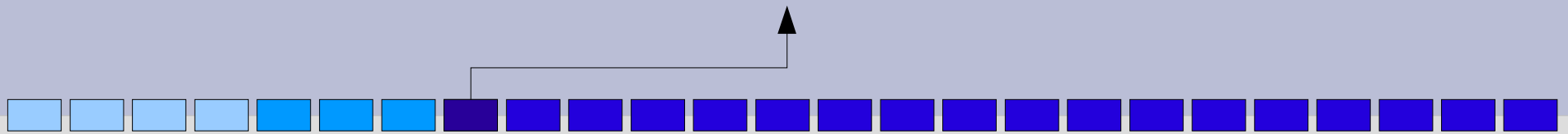
- mechanischer Wecker
- Der Wecker nimmt die gezeigte Zeit und führt bei Erkennung das vorgegebene Programm aus

PBE: Beispiele (2/2)

- Programme anpassen, z.B.:
  - Dialoge immer gleich beantworten
  - Auswahl festlegen
- Wiederholende Arbeiten, wie z.B.:
  - Grösse von Vierecken anpassen
  - Synchronisation PDA <-> PC
- Kleine Programme schreiben:
  - Programme kombinieren
  - Menüvorschlag erstellen

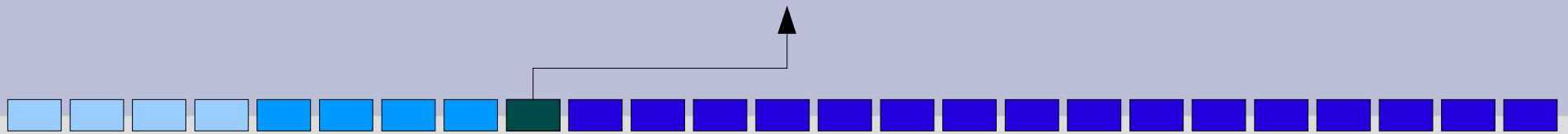


PBE: Schwierigkeiten



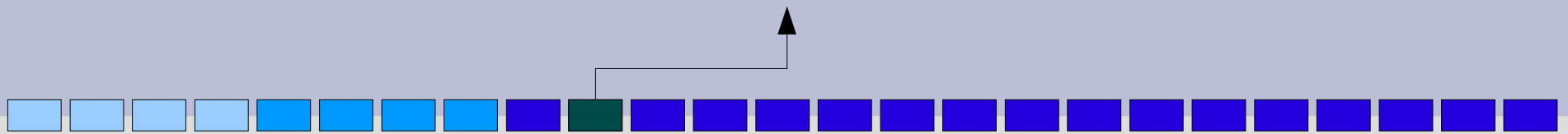
- Absicht des Benutzers muss erkannt werden
- Entscheidungsgrundlage erkennen
  - oft visuell oder im Kopf des Benutzers
- Korrekte Verallgemeinerungen finden
  - Eine Handlung kann mehrere Zwecke und Bedeutungen/Ziele haben

„a CAPpella“: Übersicht



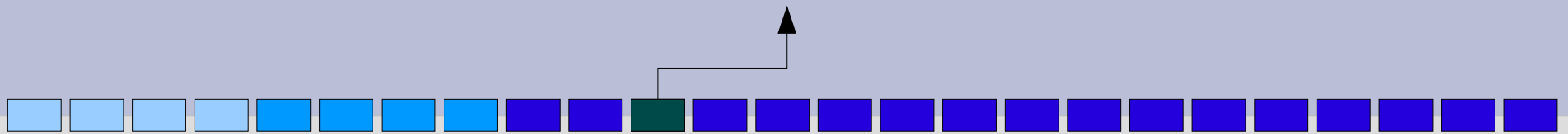
- Ziel / Ansatz
- Beispiel: Meeting
- Design / Komponenten
  - Recording System
  - Event detection
  - User interface
  - Machine learning system
- Auswertung: Fallstudie / Benutzerstudie

„a CAPpella“: Was ist das?



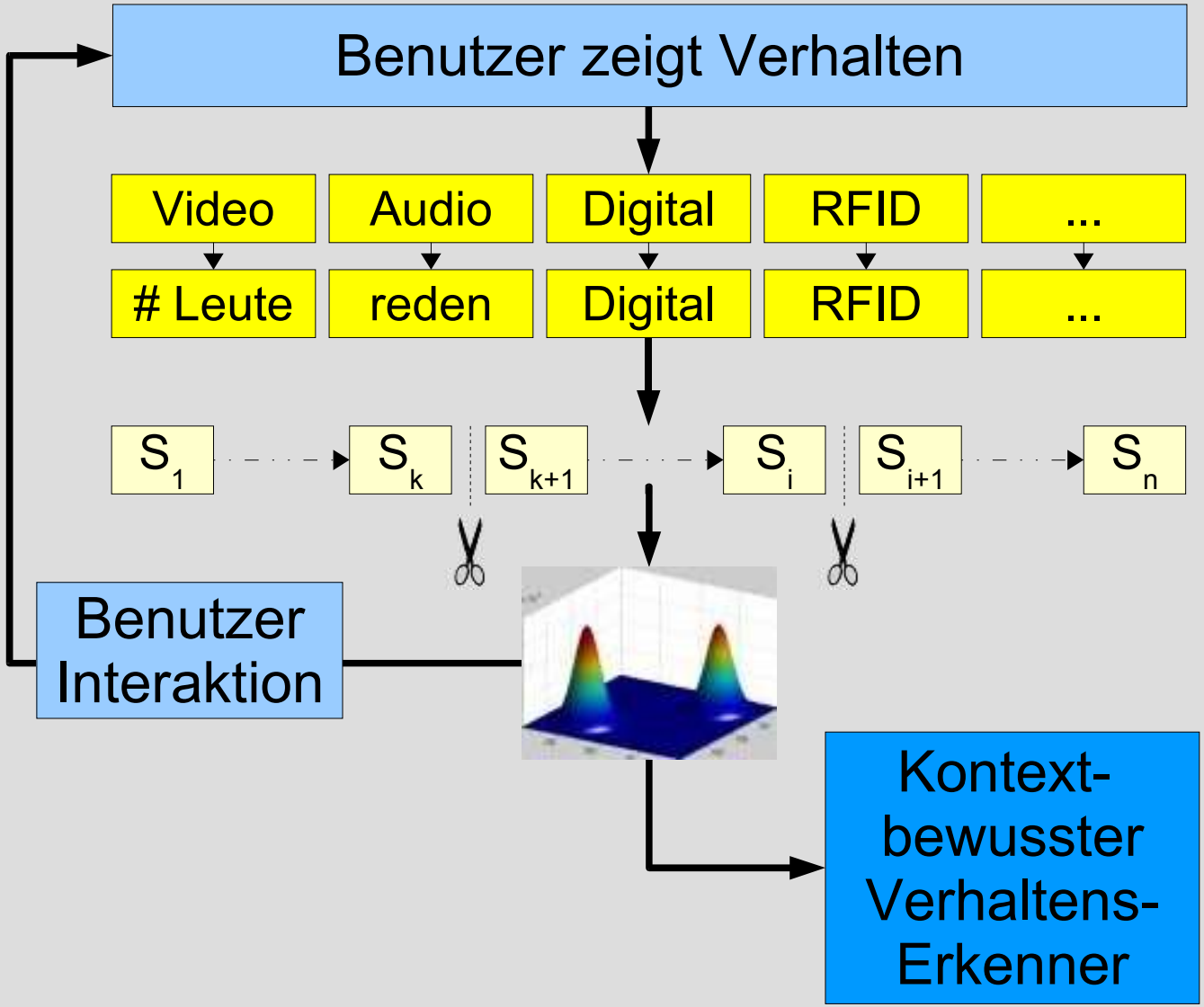
- **C**ontext-**A**ware **P**rototyping environment
- Benutzer steht im Zentrum
  - Er soll System konfigurieren/einrichten können
  - Lernaufwand soll klein sein
  - Es muss kein Code eingegeben werden
  - Konfigurationsaufwand soll klein sein
- System soll möglichst breit einsetzbar sein
- System soll sich dynamisch den Änderungen anpassen können

„a CAPpella“: Context-awareness



- Wichtig in ubiquitären Umgebungen
- Viele Entwicklungen von Tools und Anwendungen vorhanden
- Zwei Lösungs-Ansätze und deren Probleme:
  - *Regelbasiert* (ohne künstliche Intelligenz): Regeln sind oft schwierig zu definieren
  - *Erkennungsbasiert* (mit künstlicher Intelligenz): Kompliziert und schwierig zu konfigurieren

„a CAPpella“: Funktionsweise



System sammelt Sensor-Daten und erkennt Ereignisse

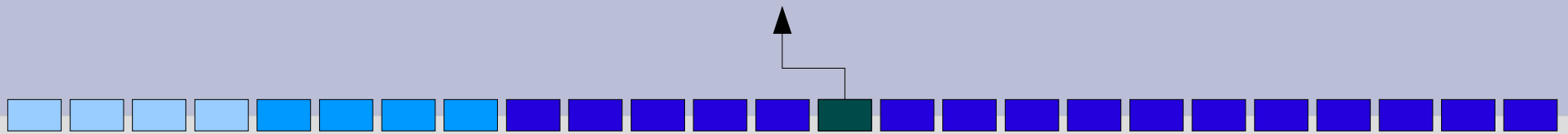
Benutzer beschreibt Daten

System trainiert Modell mit diesen Daten

„a CAPpella“: Beispiel (1/2)

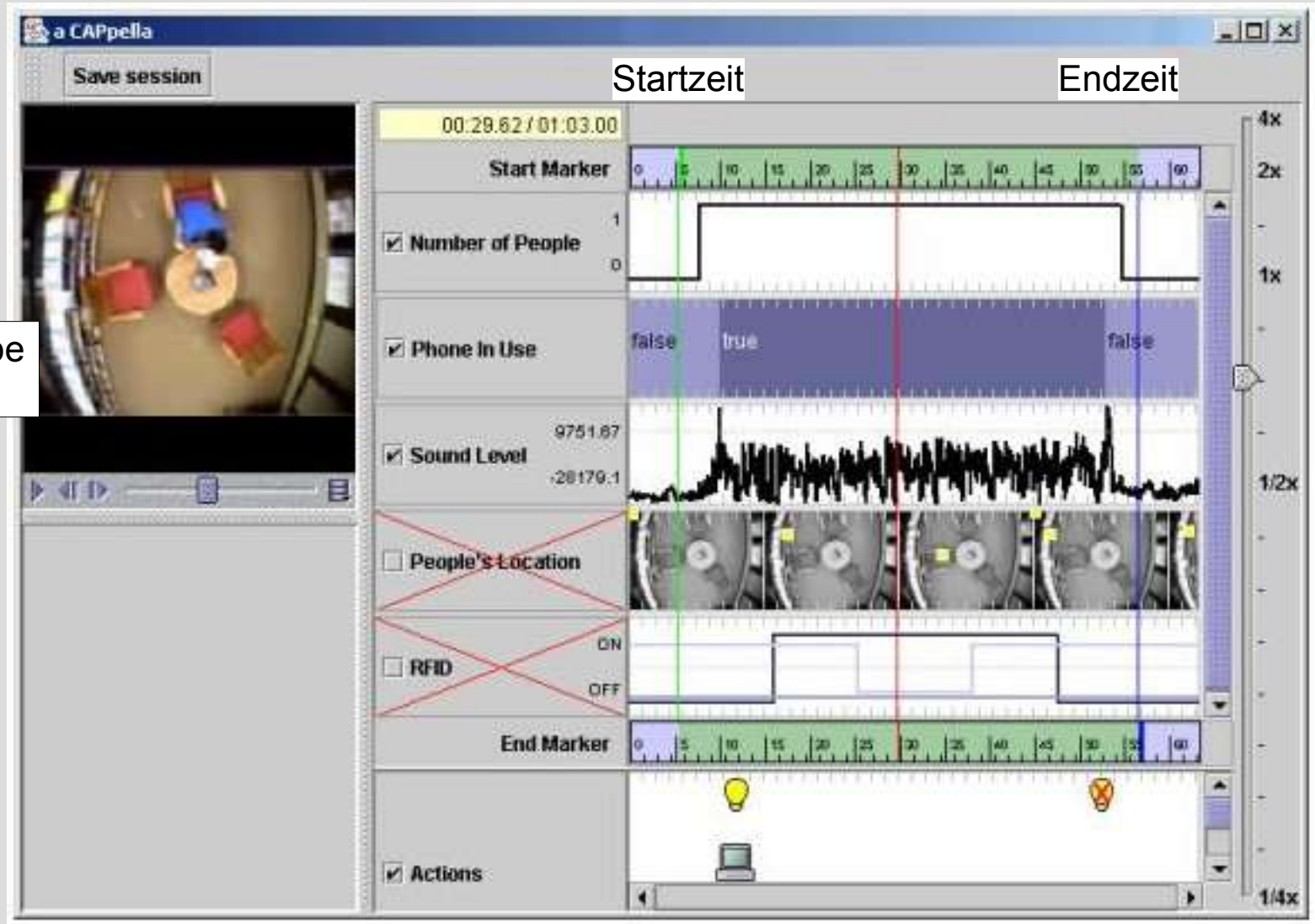
- Meeting-Situation ist:
  - schwierig zu beschreiben
  - einfach zu erkennen
- starten des „a CAPpella“ Aufnahme-Systems
  - nimmt alle Daten von allen Sensoren auf
- Meeting durchführen, mit allen Handlungen, die das System übernehmen soll
- Meeting beendet → stoppen der Aufnahme

„a CAPpella“: Beispiel (2/2)



- Wechsel auf die Benutzeroberfläche
- Benutzer beschreibt die Daten
  - Auswahl der relevanten Datenströme
  - Setzen von Anfangs- und Endzeit der Handlung
- Wiederholungen, um Resultat zu verbessern
- Danach läuft das System:
  - Erkennt die programmierte Situation →
  - Führt die dazu passenden Befehle aus

„a CAPpella“: Benutzer-Oberfläche

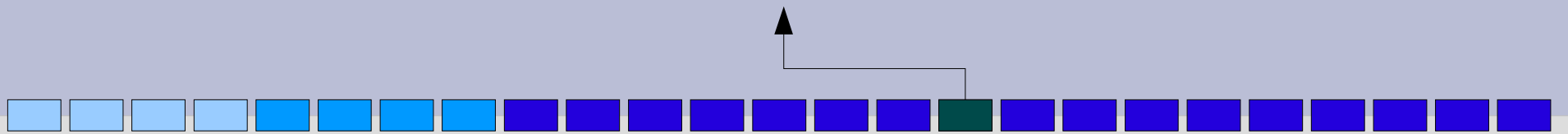


Wiedergabe Fenster

Ereignis Fenster

Action Fenster

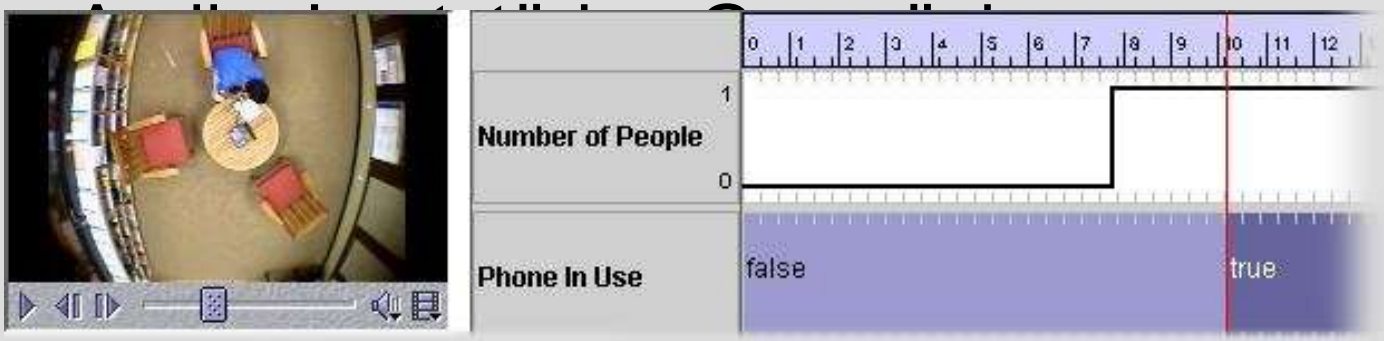
„a CAPpella“: Aufnahme-System



- Zeichnet mehrere Sensoren gleichzeitig auf
- Aufnahme von Situation und Handlung
- Sensoren:
  - Overhead-Video-Kamera
  - Mikrofon
  - RFID-Antennen
  - Telefon-Sensor (benutzt oder nicht)
  - Licht-Schalter
  - Computer (benutzte Programme, ...)
  - weitere Sensoren können hinzugefügt werden

„a CAPpella“: Ereignis-Erkennung

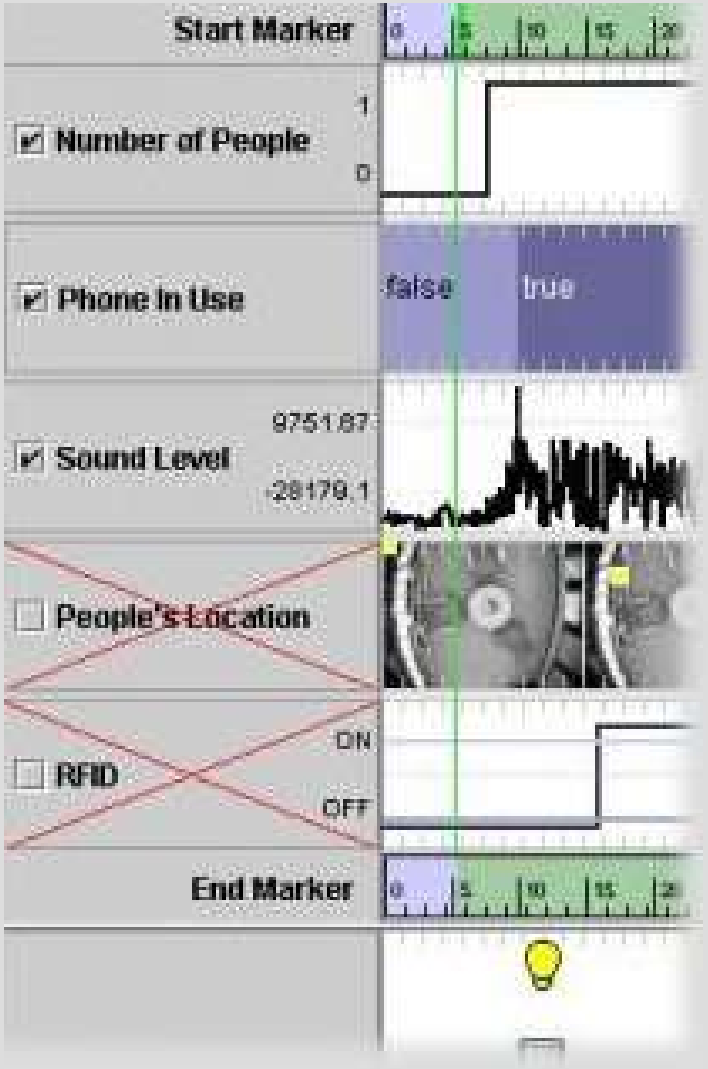
- Ableiten von „höheren“ Ereignissen aus „Rohdaten“
  - Video: Anzahl Personen, deren Position im Raum



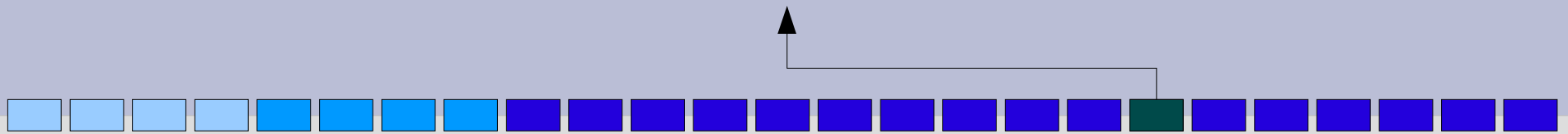
- Zusätzliche Machine Learing Algorithmen

„a CAPpella“: Benutzer-Oberfläche

- Einfache und intuitive Benutzung steht im Vordergrund
- Mehrmals getestet
- Darstellung ver-schiedener Datentypen
  - Boolean, Ganzzahlen, Kommazahlen
  - Ort, RFID, Ereignisse

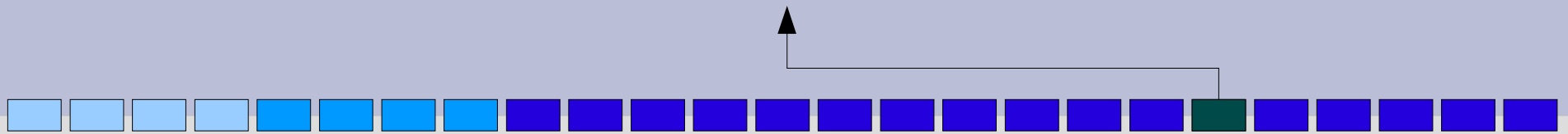


„a CAPpella“: Machine Learning System



- Dynamic Bayesian Network
  - Hidden Markov Model
- Zwei Modelle für jede gezeigte Aktivität
  - z.B. Meeting findet statt / findet nicht statt
- Modelle werden mit Testdaten verglichen und das Modell mit höchster Wahrscheinlichkeit wird ausgewählt
- Verglichen wird periodisch (1 x pro Sekunde)
  - Sliding Window von 10 Sekunden
  - kurze Verzögerung (Verbesserungsmöglichkeit)
- Recognizer verschieden einsetzbar

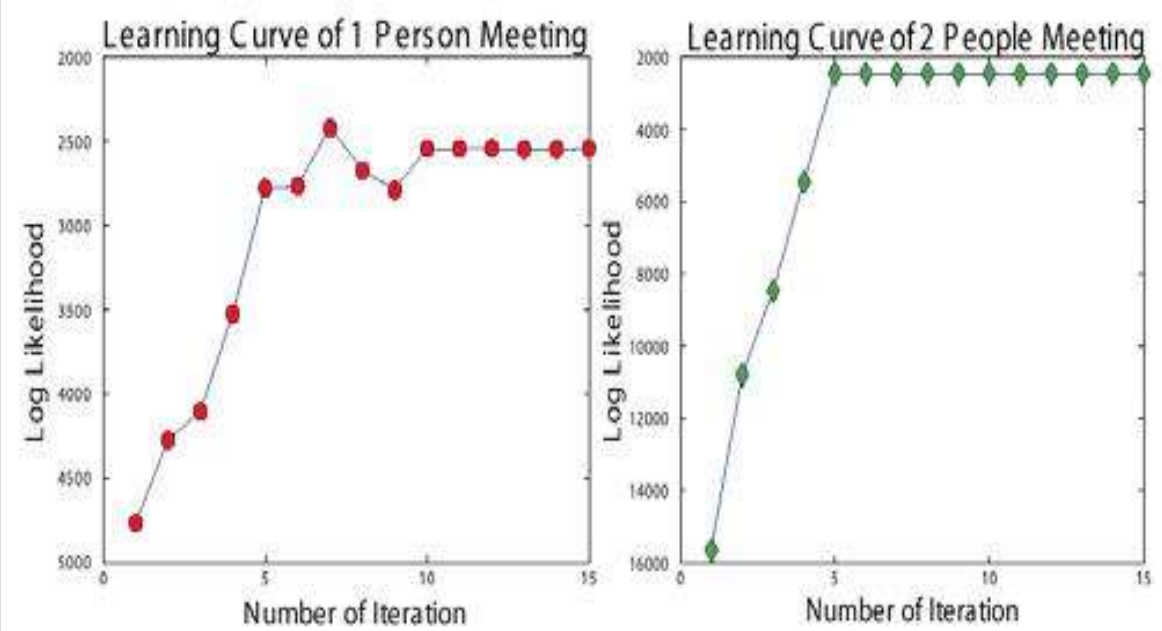
„a CAPpella“: Fallstudie (Entwickler)



- Meeting Szene
  - Daten von Meetings mit 1 (Telefon-Meeting) oder 2 Personen
  - Daten von 1 oder 2 Personen, die kein Meeting haben
- Entwickler benutzen das System
- Komplexere Benutzer-Oberfläche
- 15 Meetings für Modellanpassung
- Brauchbarkeit des Systems soll gezeigt werden

„a CAPpella“: Resultate (Entwickler)

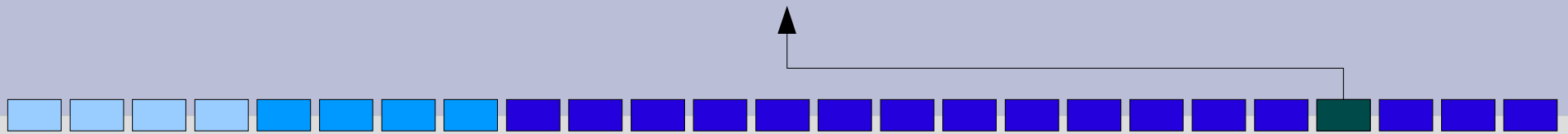
Training:



Test:

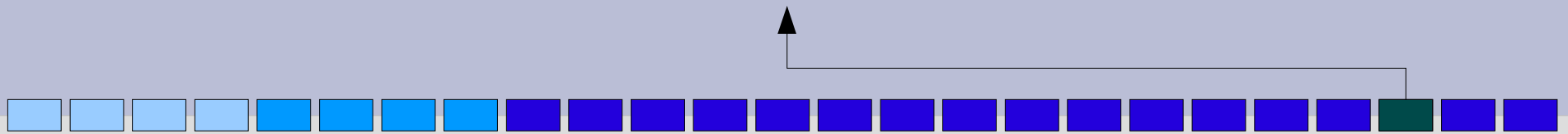
	1P M	1P NM	2P M	2P NM
1P M	93.30%	6.60%	0.00%	0.00%
1P NM	13.30%	86.60%	0.00%	0.00%
2P M	0.00%	0.00%	80.00%	20.00%
2P NM	0.00%	6.60%	6.60%	86.60%

„a CAPpella“: Benutzer-Studie (1/2)



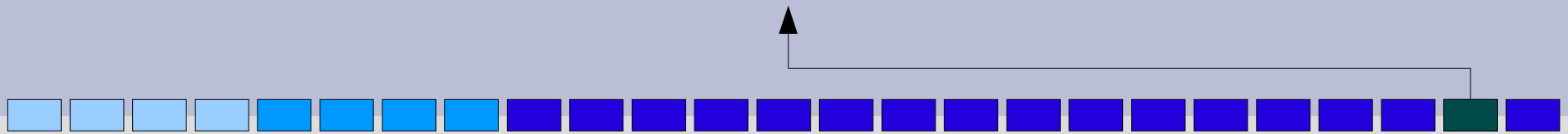
- Benutzer:
  - 14 Teilnehmer, keine Informatiker
  - 18 – 60 Jahre alt
  - 5 Minuten Einführung in die System-Benutzung
- Meeting Szenen
  - 3 x 1-Personen Telefon-Meetings,
    3 x 2-Personen Meetings
  - Benutzer mussten relevante Kanäle (z.B. Anzahl Personen, Lautstärke) und Zeit (Start, Ende) selbst auswählen

„a CAPpella“: Benutzer-Studie (2/2)



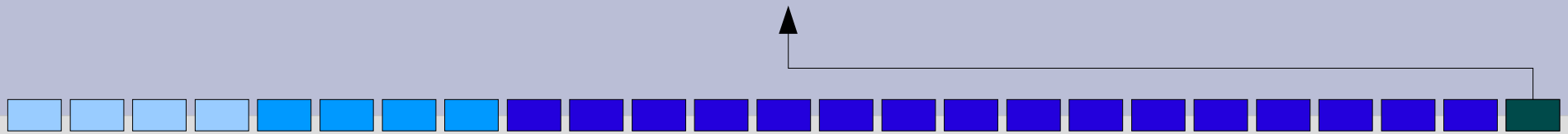
- Resultat:
  - 1-P Meeting/kein Meeting:
    - 67.2% richtig erkannt (zwischen 59.5% und 73.3%)
  - 2-P Meeting/kein Meeting:
    - 55.5% richtig erkannt (zwischen 50.0% und 78.6%)
  - nicht vergleichbar:
    - Anzahl Versuche ist verschieden
    - Andere Benutzeroberfläche

Ausblick



- Negativ
  - Resultat der Benutzer ist schlecht
  - Reihenfolge wird vernachlässigt
  - Ziemlich aufwendig (min. 5 Wiederholungen)
- Unklar
  - Was passiert in ähnlichen Situationen (Essen/Meeting)?
  - Wie kompliziert dürfen die Probleme sein?
- Positiv
  - natürlicher/intuitiver Programmierstil
  - Bedarf in ubiquitären Umgebungen ist da

Fragen?



- Referenzen:

  - Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, Daniel Hsu
    a CAPpella: Programming by Demonstration of Context-Aware Applications.
    Proceedings of CHI 2004, Vienna, Austria, April 2004

  - Allen Cypher (Ed.)
    Watch What I Do – Programming by Demonstration.
    MIT Press, Cambridge, MA, USA, 1993

  - Henry Lieberman (Ed.)
    Your Wish is my Command: Programming by Example.
    Academic Press, 2001