XPerto

Voice AI Agent für Meetings

Ein Projekt von
B.A. Lena Leybovich
M.Eng. Jason Schühlein

Betreut durch
Prof. Dr. David Klotz

HOCHSCHULE DER MEDIEN

Agenda

1. Projektvorstellung
2. Grundlagen Voice Agents
3. Implementierung von XPerto
4. Methodik und Auswertung
5. Ausblick
6. Live Demo

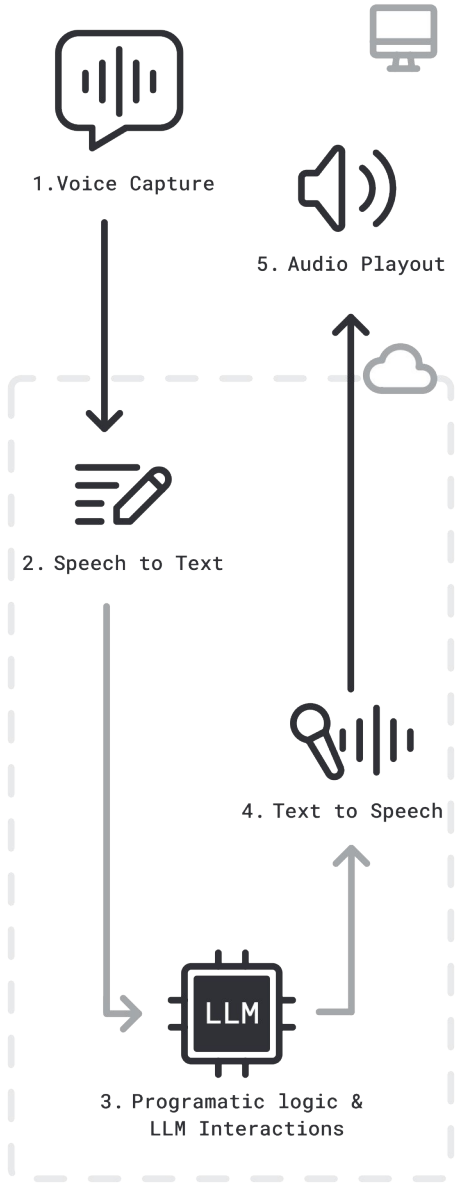
Projektvorstellung XPerto

- Projekt im Rahmen des **Master of Media Research**
- Kollaboratives Szenario: KI als Fachexperte für Meetings
- Unterstützt durch domänenspezifische Expertise
- Expertise in Bereichen wie Recht, Sicherheit und Finanzen
- Erkenntnisse über Akzeptanz von kollaborativen Systemen

Anforderungen an XPerto

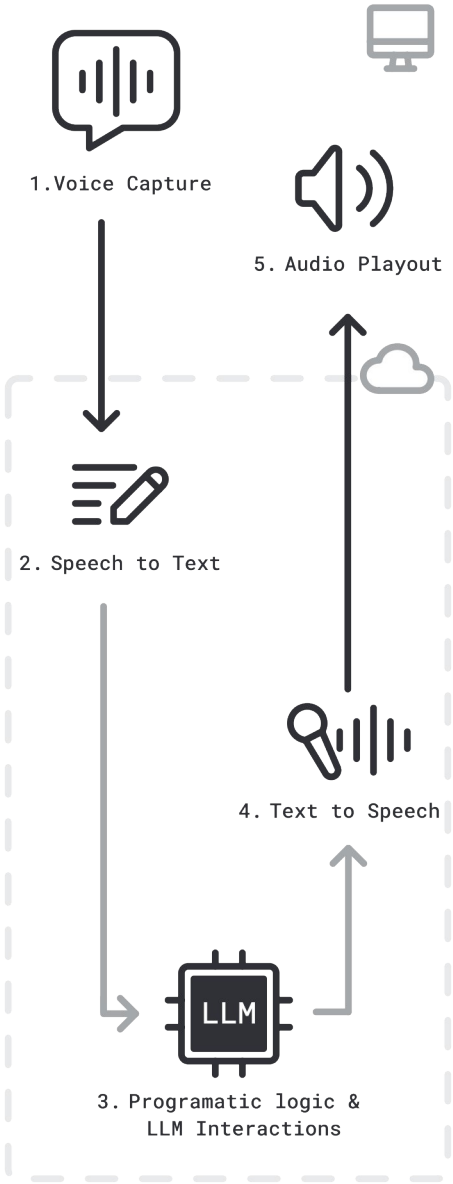
- Kleinere bis mittlere Meetings mit 2 – 10 Teilnehmern
- Sprachen: Deutsch und Englisch
- Konfigurierbare Persona und Rolle des Agenten
- Kontextverständnis und domänenspezifische Expertise
- Echtzeit-Antworten und Hinweis auf weiterführende Quellen
- Einbindung ins Meeting (Audio/Chat/Visualisierung)
- Kriterien: Transparenz, Erklärbarkeit, Vertrauen
- Ethik & Recht: Datenschutz, Transparenzpflicht, Risikobewertung, Human-Control

Grundlagen Voice Agents



Die Voice Agent Pipeline

- **Speech-To-Text (STT)**
  - ML Modell transkribiert Sprache
- **LLM Agent**
  - Gewöhnliches LLM (GPT4o)
  - System Prompt und Persona
  - Potentiell auch Tools
- **Text-To-Speech (TTS)**
  - ML Modell synthetisiert Sprache aus Text



Turn Detection

- *Woher wissen wir wann jemand fertig ist mit Sprechen und eine Antwort erwartet?*
- **Endpoint Marker** (Walkie-Talkie)
  - Etwas gewöhnungsbedürftig, Over.

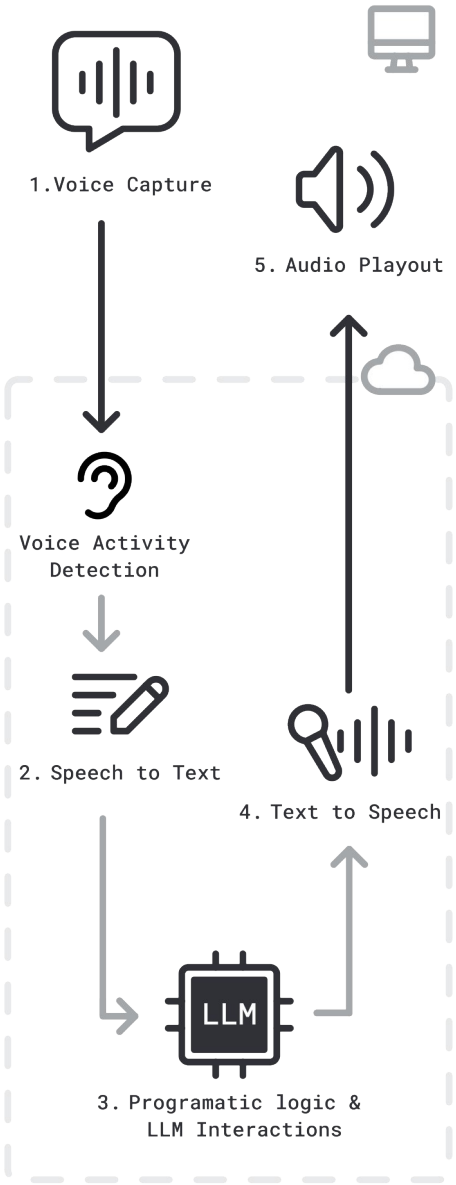
- **Push-To-Talk**
  - Eindeutigste Option
  - Unnatürliche User Experience

Turn Detection

- *Woher wissen wir wann jemand fertig ist mit Sprechen und eine Antwort erwartet?*
- **Voice Activity Detection (VAD)**
  - Erkennt ob gerade jemand spricht
  - Naiver Ansatz: Lautstärke Threshold
  - Länge der Sprechpausen messen und daran entscheiden
  - ML Ansatz: Modell klassifiziert Audio in Sprache und Umgebungsgeräusche
- **Context-Aware Turn Detection (Semantic VAD)**
  - Erkennung anhand Füllwörter, Semantik, Grammatik und Aussprache
  - Deep Learning Modelle wie „Smart Turn“

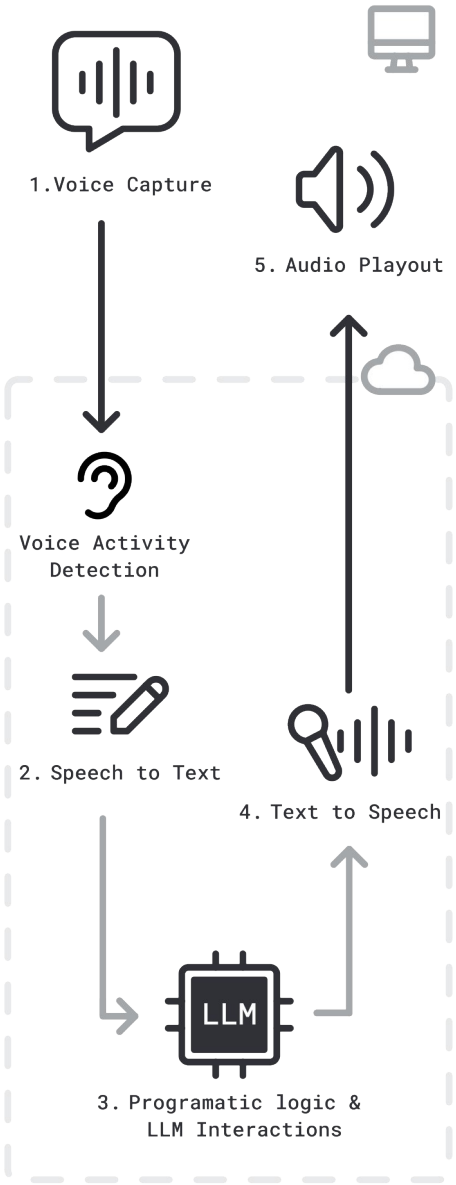
Die Voice Agent Pipeline

- **Speech-To-Text (STT)**
  - ML Modell transkribiert Sprache
- **LLM Agent**
  - Gewöhnliches LLM (GPT4o)
  - System Prompt und Persona
  - Potentiell auch Tools
- **Text-To-Speech (TTS)**
  - ML Modell synthetisiert Sprache aus Text



Unterbrechungen

- *Wie funktionieren Unterbrechungen eines Voice Bots?*
- **Voice Activity Detection (VAD)**
  - Signal sobald jemand redet
  - Unterbrechung des TTS Modells
  - Mindest-Sprechdauer um „Ok“, „Mhm“, etc. rauszufiltern
- **Vorsicht vor Kontext Drift**
  - Nur das Gesprochene sollte im Transkript stehen
  - Word Level Timestamps

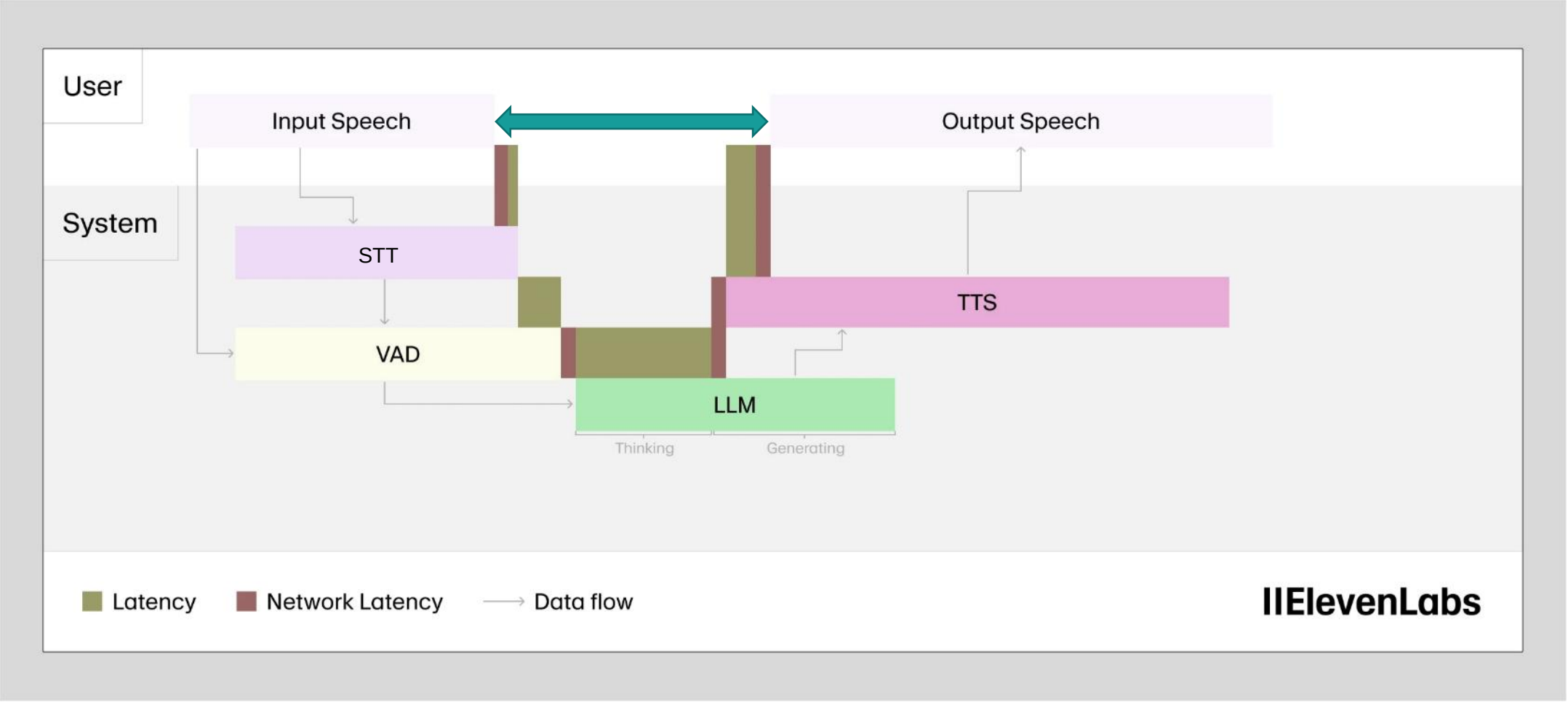


Latenz

- Menschliche Konversation: ~ 500 ms
  - Zu lange Pausen wirken unnatürlich

- Übliches Ziel für Voice AI: < 800 ms
  - Lokales vs. Cloud Deployment
  - Trade-off zwischen Qualität und Latenz

Latenz



https://elevenlabs.io/blog/how-do-you-optimize-latency-for-conversational-ai

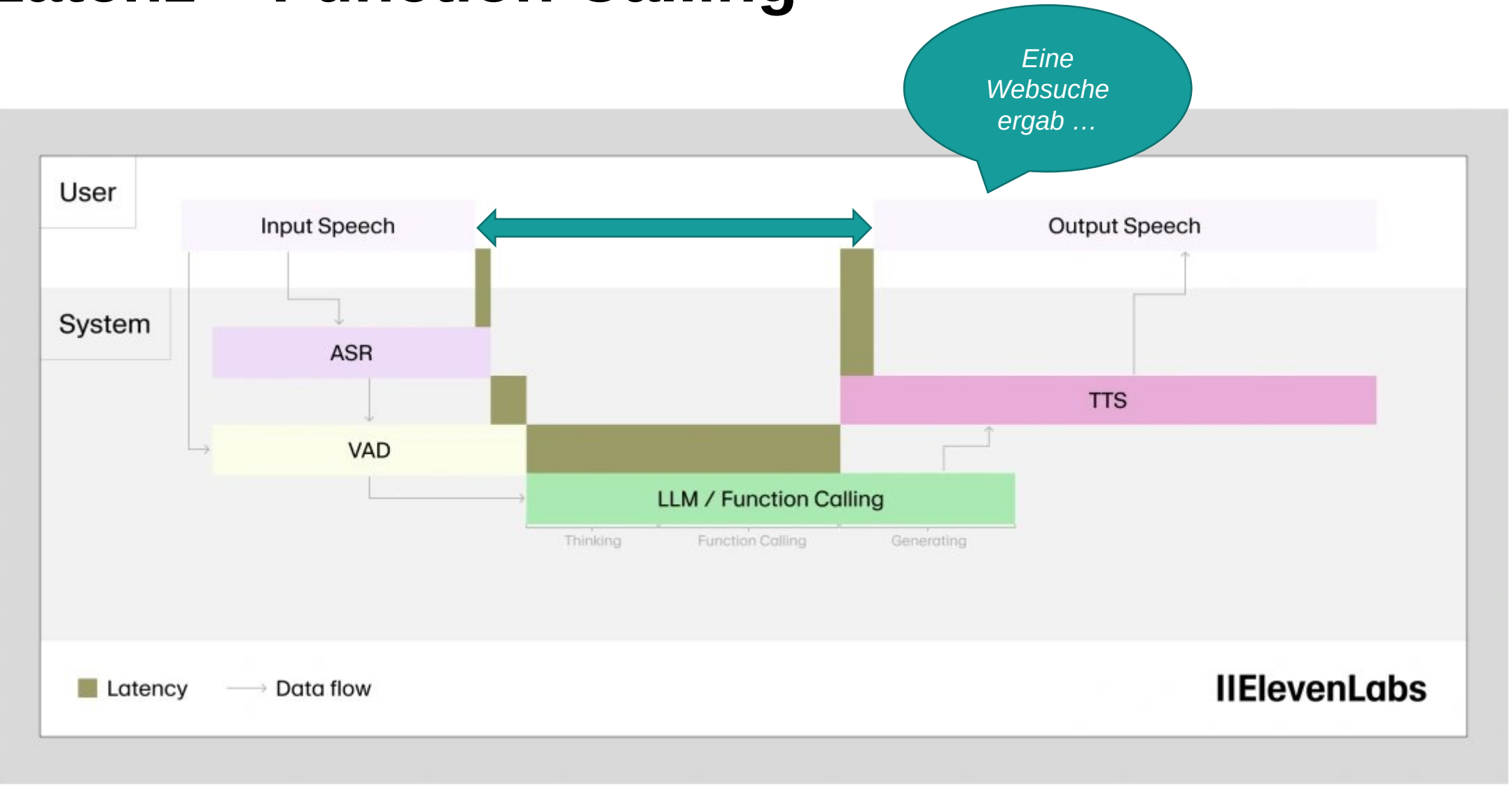
Function Calling

- Viele Voice Agents dienen als **Interface**
  - Essensbestellung, Call Center, …
  - Interaktion mit anderen Services
  - Potentiell nicht auf Echtzeit ausgelegt

- Abfrage von fehlenden Informationen
  - Web Suche und Zusammenfassung
  - Retrieval aus Dokumenten

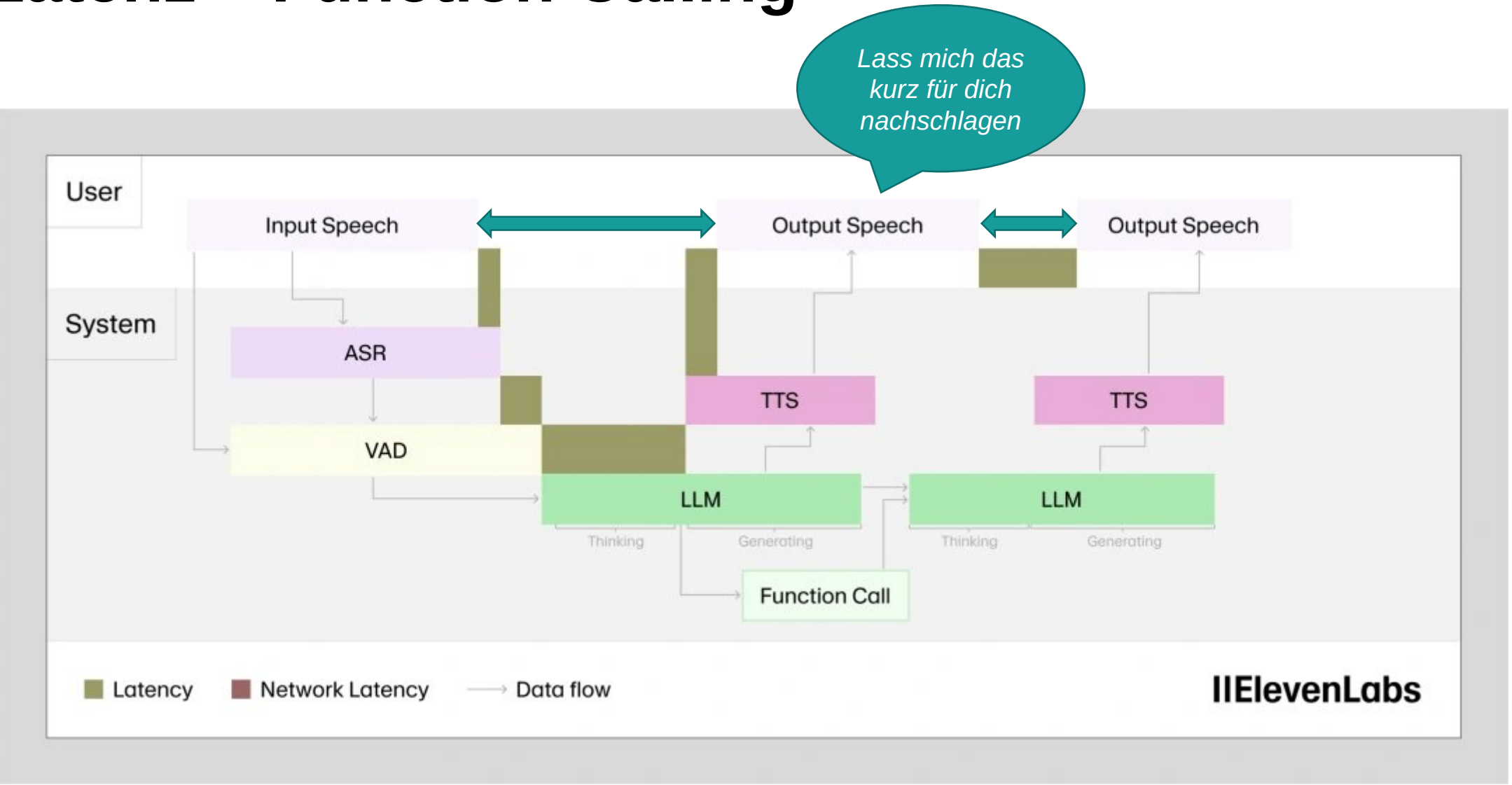
- Fügt weitere Latenz hinzu

Latenz – Function Calling



https://elevenlabs.io/blog/how-do-you-optimize-latency-for-conversational-ai

Latenz – Function Calling



https://elevenlabs.io/blog/how-do-you-optimize-latency-for-conversational-ai

Speech To Speech?

- OpenAI Realtime API (`gpt4o-audio-preview`)
- **Vorteile**
  - Geringe Latenz
  - Verständnis von Intonation des Users (Emotion, Frage)
  - Natürlichere Sprache
- **Nachteile**
  - Noch recht neu, wird aber immer besser
  - Weniger Kontrolle über Inputs/Outputs (Guardrails)
  - Audio benötigt mehr Token als Text (~ Faktor 10)
  - Höhere Kosten und kürzere Konversationen

Building XPerto

- Problemstellung Meeting
  - Keine 1:1 Konversation mehr
  - Turn Taking in Gruppen
  - Unterscheidung von Sprechern
- Implementierung
  - Pipecat Framework
  - Verwendete Services / Modelle

Transkription bei mehreren Sprechern

- Transkript unterscheidet aktuell nur zwischen „User“ und „Assistant“

- Option 1: **Channel Diarization**
  - Jeder Sprecher hat ein eigenes Mikrofon
  - Gegeben im Falle von Online Meetings

- Option 2: STT mit **Speaker Diarization**
  - Erkennung von Sprechern anhand Stimm-Muster
  - STT Modell weist Sprecher ID zu (mit Confidence Score)
  - Unterscheidung Vordergrund- und Hintergrundsprecher

Speaker Diarization

- Markierung der Sprecher für das LLM mittels Sprecher-Tags:

```
user: <S1>Hallo Experto, Mein Name ist Jason.</S1>
user: <S2>Hallo. Ich bin Lena.</S2>
assistant: Danke Lena. Seid ihr nur zu zweit oder fehlt noch jemand?
```

Turn Detection

- *Woher wissen wir wann die Gruppe eine Antwort vom Bot erwartet?*
- **(Semantic) VAD**
  - Geht davon aus dass der Bot dran ist sobald Sprecher fertig ist
  - Potenziell den Schwellwert für die Pausenlänge erhöhen

- **Wake Phrase**
  - *„Hey ~~Siri~~ Xperto!“*
  - Simpel aber nicht sehr natürlich

Turn Detection

- *Woher wissen wir wann die Gruppe eine Antwort vom Bot erwartet?*
- **Hybrider Ansatz**
  - Wake (VAD) und Sleep (Wake Phrase) Modus
  - Bleibt wach bis er einige Zeit nicht mehr angesprochen wurde (Timeout)
  - Sobald der Name fällt wacht der Bot wieder auf

- **Proaktives Eingreifen?**
  - Unterbrechen des Meetings durch den Bot
  - Noch nicht möglich aber denkbar
  - Erfordert kontinuierliches „Denken“ (Kosten!)

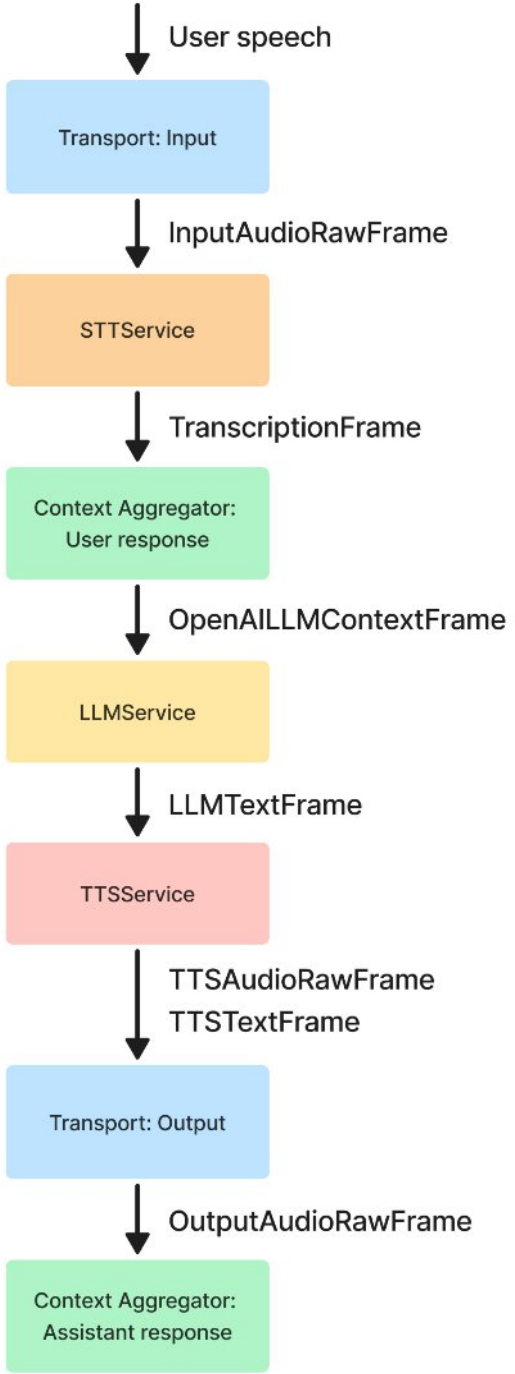
Pipecat Framework

- Python Framework für Voice Agents
  - Integrationen mit etlichen Services / APIs
  - Deployment: Lokal, Webapp und Telefonie

- **Frames:** Kleinste Pakete an Daten (Audio, Video, Text, …)
- **Frame Processors:** Spezialisierte Worker die gezielte Pakete verarbeiten
- **Pipeline:**
  - Sequenzielles Verbinden der Worker
  - Orchestriert den Fluss der Pakete
- **Parallelisiert** und **Modular**

Pipecat Processing Flow

1. **Audio Input**
   - User spricht, Audio wird als Audio-Frame gestreamt
2. **Speech Recognition**
   - STT bekommt Audio-Frame, transkribiert es und pusht ein Text-Frame
3. **Context Management**
   - Aggregiert Text-Frames und formatiert diese für das LLM
4. **Language Processing**
   - LLM bekommt Context und streamt Antwort als LLM-Frame
5. **Speech Synthesis**
   - Bekommt LLM-Frame und synthetisiert Sprache als Audio-Frame
6. **Audio Output**



Pipecat Processing Flow

```
pipeline = Pipeline([
    transport.input(),                  # Receives user audio
    stt,                                # Speech-to-text conversion
    context_aggregator.user(),          # Collect user responses
    llm,                                # Language model processing
    tts,                                # Text-to-speech conversion
    transport.output(),                 # Sends audio to user
    context_aggregator.assistant(), # Collect assistant responses
])
```

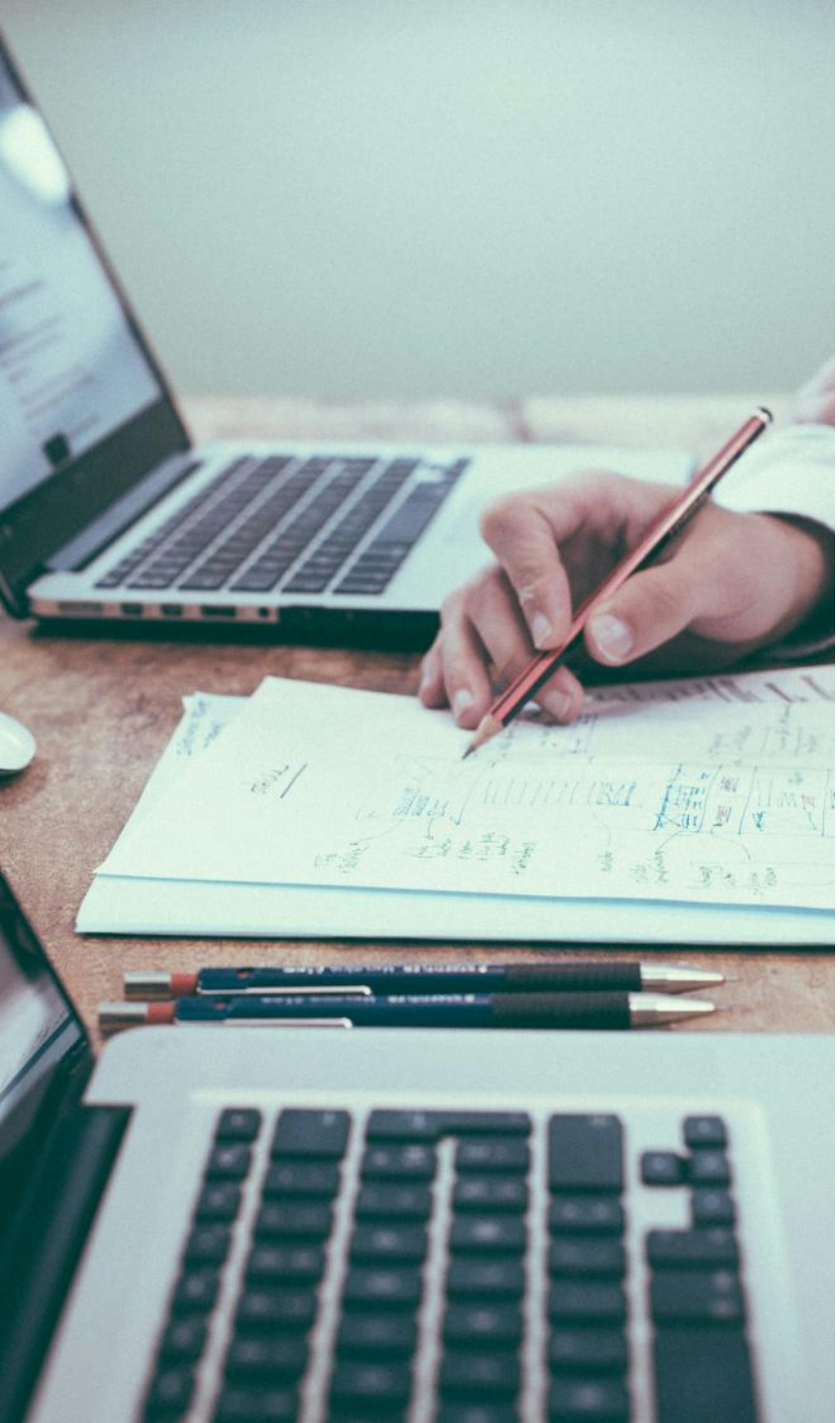
Auswahl der Services

- VAD: `SileroVAD`
  - Lokal und schnell
  - Alternativ: Smart Turn
- STT: `Speechmatics`
  - Speaker Diarization auch für Deutsch
  - Alternativ: Deepgram
- LLM: `OpenAI GPT4.1`
- TTS: `ElevenLabs Flash 2.5`

SPEECHMATICS

OpenAI

IIElevenLabs



Methodik und Auswertung

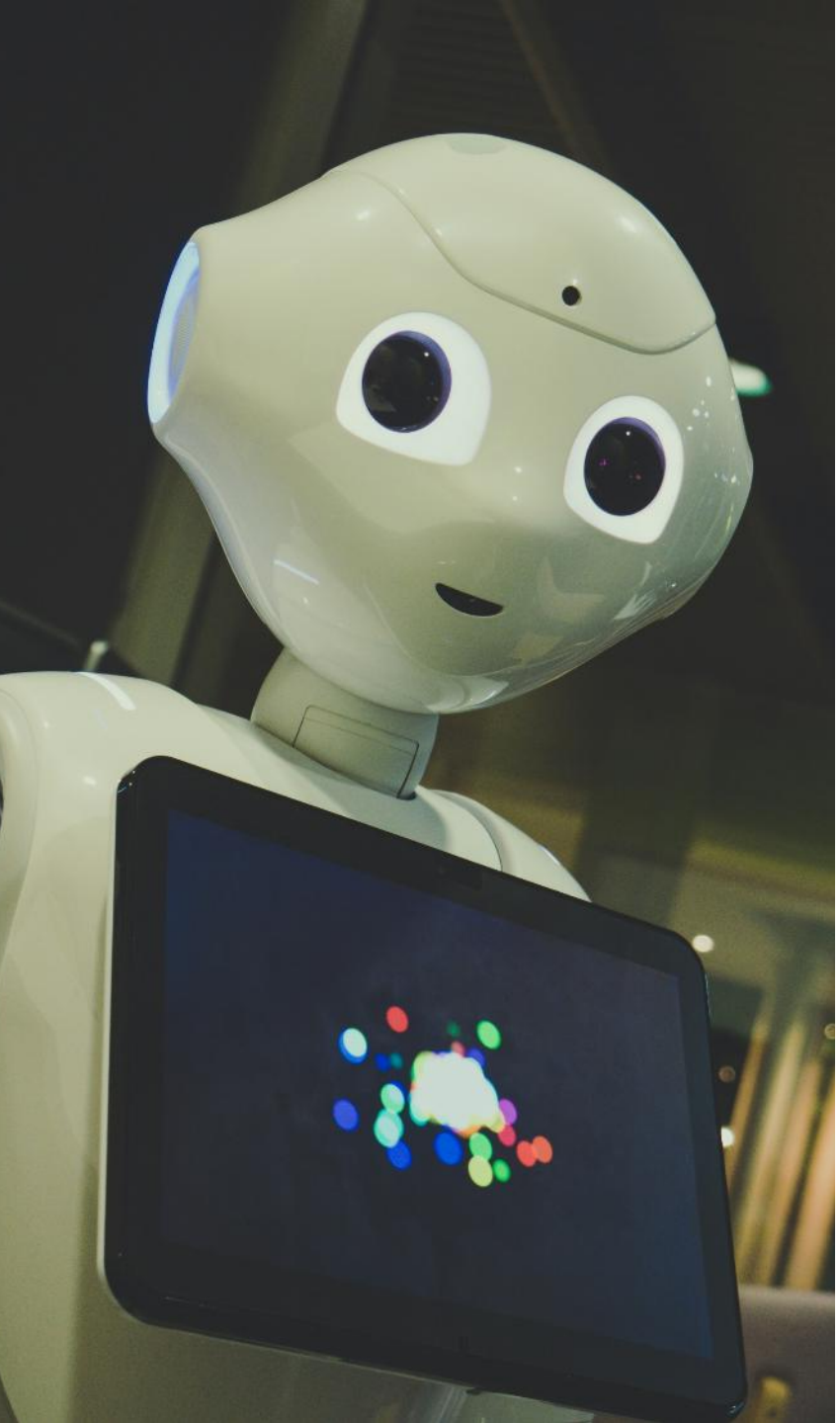
Methodik

- Interner Pretest und Evaluation des Prototyps
- Qualitative Vorstudie: realitätsnahe Meetings
- Hauptstudie mit Unternehmenspartnern

à Empirische Untersuchung im realen Anwendungs-Kontext

Auswertung

- Qualitative Inhaltsanalyse nach Mayring
- Ableitung von Designprinzipien für Akzeptanz kollaborativer Systeme



Ausblick

- Ausbau von Assistent zu Agent (RAG, WebSearch)
- Betrachtung rechtlicher und ethischer Aspekte abhängig des Anwendungsgebiets (z.B. HR)
- Glaubwürdigkeit und Bias von LLMs

Anknüpfend an diesen Vortrag:

Further Reading

- *„Voice AI & Voice Agents – An Illustrated Primer“*
  https://voiceaiandvoiceagents.com/

- *„How do you optimize latency for Conversational AI?“*
  https://elevenlabs.io/blog/how-do-you-optimize-latency-for-conversational-ai

- *„Learning Pipecat“*
  https://docs.pipecat.ai/guides/learn/

Q & A

```
> python xperto.py
```