A Modular Multimodal Dialog Architecture for Digital PROM Collection

Stefan Hillmann[1], *Philipp Harnisch*[1], *Daniel Schuhmann*[1],
Navid Ashrafi[2], *Jan-Niklas Voigt-Antons*[2]

[1]*Technische Universität Berlin,* [2]*Hochschule Hamm-Lippstadt*
stefan.hillmann@tu-berlin.de

Abstract: Patient-reported outcome measures (PROMs) are important for patient-centered healthcare, but their digital implementation still faces significant usability and interoperability barriers. In this work, we present a comprehensive overview of the technical architecture and implementation of the multimodal system that was developed in the context of the MIA-PROM project.

1 Introduction

Patient-reported outcome measures (PROMs) constitute a key component of patient-centered healthcare, yet their digital deployment continues to be hindered by substantial usability and interoperability challenges. The *MIA-PROM*[1] project seeks to address these limitations through the development of a multimodal dialog system designed to support patients during the completion of PROMs and to enable seamless integration of these measures into routine clinical workflows.

Previous publications have presented complementary aspects of this system: its conceptual and participatory design foundations [1, 2], empirical evaluations of automatic speech recognition and natural language understanding in spoken PROM interactions [3], and studies on virtual agents' visual and behavioural characteristics that inform user engagement and identification in therapeutic contexts [4].

In this work, we present a comprehensive overview of the technical architecture and implementation of the multimodal system that was developed and empirically evaluated in the context of two field studies. In the following, we first present the overall system architecture, followed by a more detailed description of the core components of the dialogue system, with particular emphasis on the underlying speech and language processing technologies. The paper concludes with a discussion of the major challenges encountered during the practical deployment and field application of the system.

2 System Architecture Overview

2.1 User Interface

From the user's perspective, the system is embodied as a tablet running the MIA-PROM application (Fig. 1, middle), which can optionally be complemented by an avatar. Two avatar variants are implemented: a physical avatar (PA; Fig. 1, left), realized as an interactive robotic head that can be positioned in close proximity to the user (e.g., on a table), and a virtual avatar (VA; Fig. 1, right), rendered on an additional tablet.

[1] `https://mia-prom.de/`, `https://web.archive.org/web/20250801000000*/https://mia-prom.de`

Figure 1 – In MIA-PROM, the system can be instantiated in three distinct configurations: (i) a standalone tablet interface (center), (ii) a tablet interface accompanied by a physical avatar (left), and (iii) a tablet interface accompanied by a virtual avatar (right).

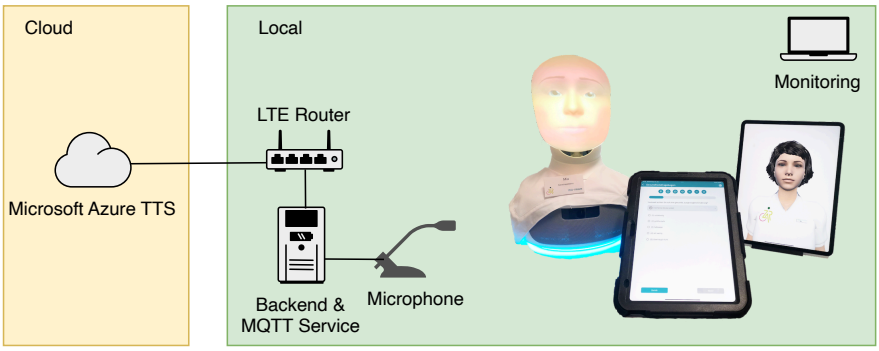

Figure 2 – Overview of the hardware configuration and its interconnection via a network router, which provided Internet access through a mobile broadband connection (LTE/5G). All devices except the backend server were connected to the router via Wi-Fi. External network access was needed to access the TTS API of the Microsoft Azure cloud and facilitated system maintenance.

User–system interaction can be conducted exclusively via the graphical user interface (GUI) of the tablet application, exclusively via speech (for both input and output), or through a multimodal combination of GUI and speech. Both avatar types can be employed as supplementary output channels for synthesized speech and as indicators of the system's listening state. The system can, however, also be operated in the absence of any avatar. Multimodal input is supported; however, MIA-PROM does not implement any modality fusion mechanisms. Instead, within a given dialog turn, the system exclusively processes the modality that is recognized first.

In the field study, all three configurations (tablet app only, tablet app with PA, and tablet app with VA) were deployed to systematically investigate the impact of different system embodiments on user–system interaction.

As shown in Fig. 2, the user's speech was captured using a conference microphone (Jabra Speak2 75) equipped with a small microphone array that supports beamforming. The microphone is directly connected to a computer running the Automatic Speech Recognition (ASR) system to minimize latency and enable direct control of the microphone (mute/unmute). In the field-test configuration, of all components depicted in Fig. 2, only the tablet running the application, the microphone, and, when applicable, the avatar were visible to the user.

2.2 Backend Components

With the exception of the TTS component (for all abbreviations used here see the caption of Fig. 3), all backend modules of the dialog system are executed on a compact workstation (Intel Core i5-12500 CPU, 32 GB RAM, RTX3060 GPU with 12 GB RAM), labeled *Backend & MQTT Service* in Fig. 2. A central element of the overall architecture is the tablet application, which not only provides the graphical user interface on the tablet, but also records all user responses to the PROM questionnaire. In this setup, the dialog state is continuously updated and processed by the DM, whereas the state of the questionnaire is managed locally by the tablet application and is transmitted to the DM only after each modification (i.e., when a question has been answered).

All backend modules are independent parallel processes on the backend server. Nevertheless, they collectively constitute a canonical dialog system pipeline, as illustrated in Fig. 4. MQTT serves as the communication protocol among the modules, the tablet application, and the avatar. This architecture is described in more detail in the following section.

2.3 MQTT for Component Communication

The central element of our system architecture is an MQTT (Message Queueing Telemetry Transport) broker, implemented as a service in accordance with the MQTT V3.1 specification[2]. As illustrated in Fig. 3, no system module or component (hereafter referred to as an MQTT client) communicates directly with any other; instead, all inter-module communication is mediated via the MQTT broker.

Each MQTT client establishes a connection to the broker through a lightweight network-based handshake. A fundamental concept in MQTT is the notion of topics. A client subscribes to those topics that are relevant to its functionality and can subsequently publish messages to, or receive messages from, these topics. For instance, both the ASR and NLU modules subscribe to the topic *ASR result* (see the top of Fig. 3). Whenever the ASR module generates a new recognition result, it publishes this output to the *ASR result* topic. The MQTT broker then automatically notifies the NLU module by delivering the corresponding MQTT message. The NLU module processes the received transcribed text and, in turn, publishes its output to the associated topic.

In addition to this illustrative example, Tab. 1 summarizes the set of topics defined for the MIA-PROM system. For reasons of clarity and readability in this paper, the set of topics presented constitutes a simplified abstraction of the more extensive hierarchy of topics and subtopics employed in the actual MIA-PROM implementation.

There are three principal reasons for employing MQTT in MIA-PROM. First, MQTT facilitates seamless communication among modules and components implemented on heterogeneous platforms and in different programming languages. MQTT client libraries are available for a wide range of programming languages. Central to the effective use of MQTT is the specification of a common message format, which must be implemented consistently in each client subscribing to a given topic.

Second, all components and modules can be implemented independently of one another with respect to both code base and runtime environment. This decoupling is advantageous during development, testing, and deployment.

Consequently, third, the implemented modules can be easily replaced by alternative versions or different implementation approaches, and they can be deployed on various physical or virtual machines, provided they can connect to the MQTT broker via a network connection.

[2] `https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html`

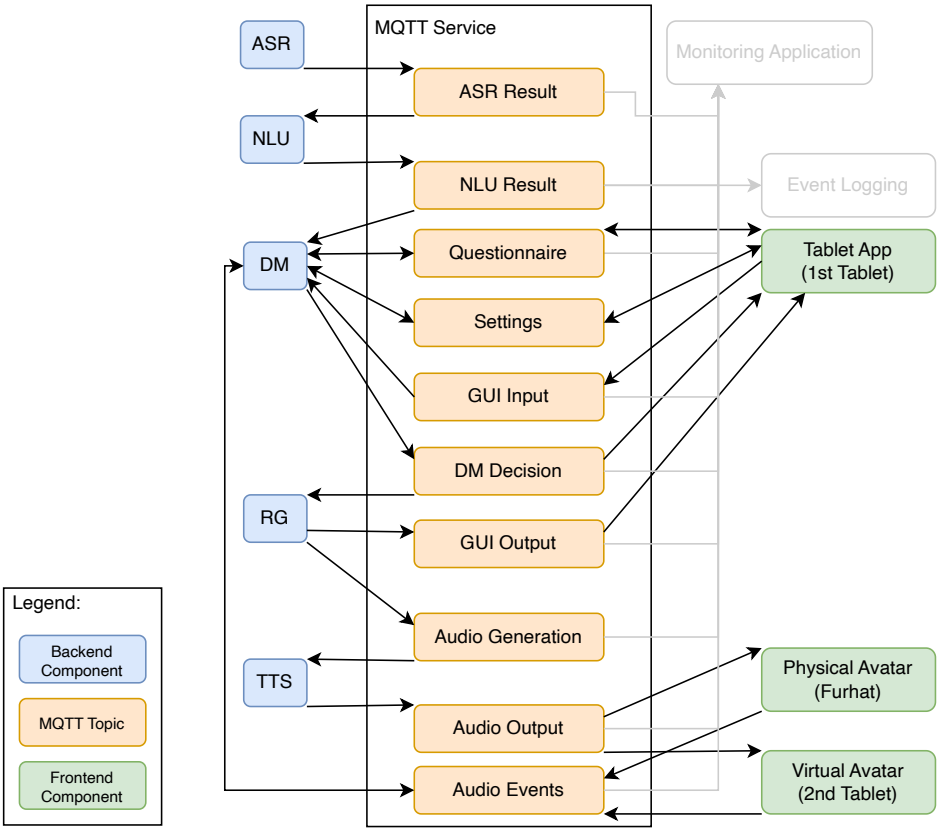

Figure 3 – Overview of the communication flow between backend services (depicted in blue) and user interface devices (frontend components depicted in green), mediated via publication and subscription to topics (depicted in orange) using the MQTT messaging protocol. Arrows directed toward a topic denote related events that are published by a given component, whereas arrows originating from a topic denote related events that are consumed by subscribing components.
Abbreviations: ASR - Automatic Speech Recognition, NLU - Natural Language Understanding, DM - Dialog Manager, RG - Response Generation, TTS - Text-to-Speech Synthesis.

However, this approach also entails drawbacks. A primary disadvantage of MQTT is the increased communication latency between modules/clients, which scales with the performance characteristics of the underlying network. This issue will be discussed for the affected modules in Sec. 3.

In addition to the backend modules and frontend components, two more modules were implemented: one for live monitoring of all backend modules and another for logging all MQTT messages for later analysis. Both are shown in gray in Fig. 3.

3 Implementation of the Core Components

3.1 Speech Processing Pipeline

Automatic Speech Recognition By default, MIA's microphone remains continuously active, enabling direct interaction without the need for push-to-talk functionality or the use of a wake word.

Our initial approach to the ASR module relied on a proprietary implementation of voice activity detection combined with OpenAI's open-source Whisper[3] models (we evaluated the small, medium, and large configurations) for speech-to-text transcription. While Whisper's word error rate (WER) proved to be adequate to very good for complete sentences, where suffi-

[3]https://github.com/openai/whisper

Topic	Description
ASR Result	Automatic speech recognition output (e.g., transcribed text accompanied by confidence scores) for subsequent downstream processing.
NLU Result	NLU results (intent and entities) for further processing.
Questionnaire	Events to control which questionnaire question is in focus, such as moving to the next/previous question or jumping to a specific one by its ID.
Settings	Changes to user-specific system settings such as speech volume or system prompt length.
GUI Input	Events triggered by interaction elements in the tablet app, such as button presses or selected questionnaire answers.
DM Decision	Decisions made by the dialog manager, i.e., system actions to perform.
GUI Output	UI elements, usually text or prompts.
Audio Generation	Text that is to be transformed into spoken language.
Audio Output	Audio (spoken language) to be played by an output device (avatar or tablet).
Audio Events	Events indicating start/stop of audio playback and mute/unmute of microphones or loudspeakers.

Table 1 – Description of the data and events assigned to the different MQTT topics. The table and Fig. 3 present a simplified version of the fine-grained topics and sub-topics taxonomy used in the implementation.

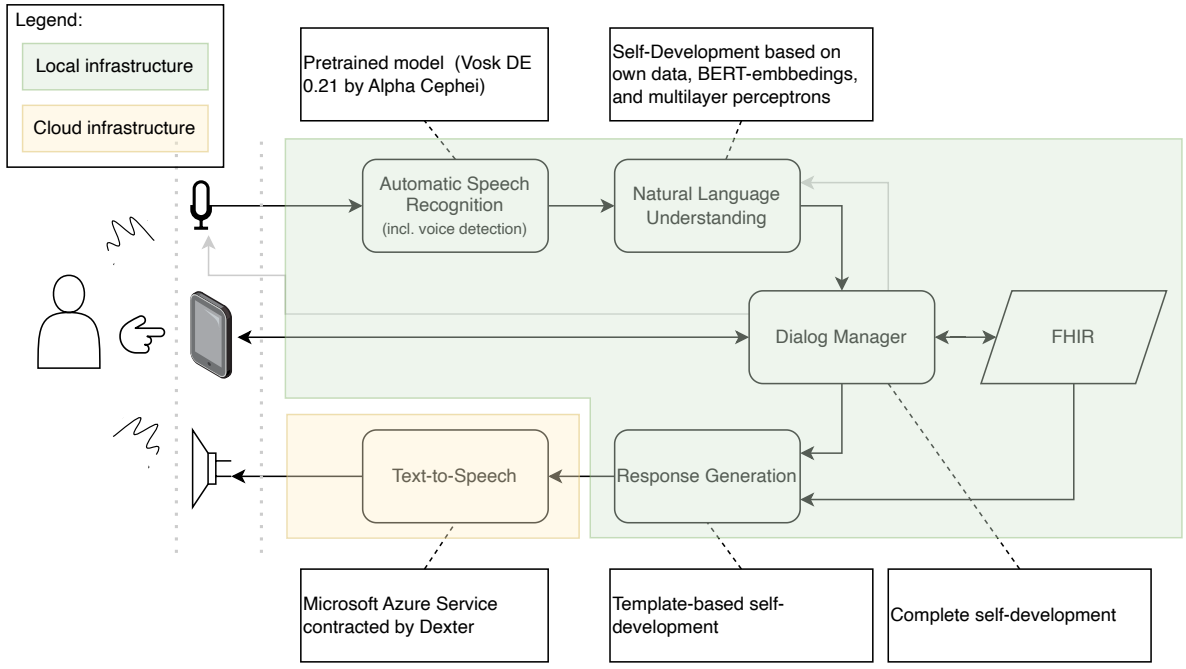

Figure 4 – Overview on the MIA-PROM speech processing pipeline. Except the text-to-speech synthesis, all components are run in a local setup without dependencies on external services (cp. Figure 2).

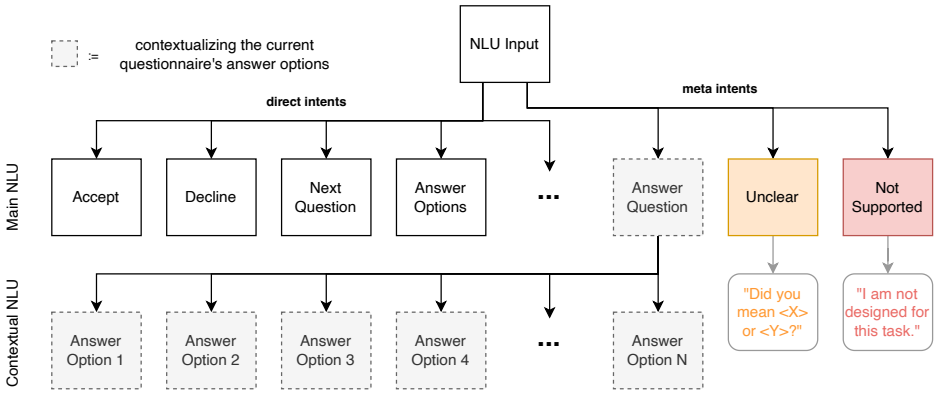

Figure 5 – Schematic illustration of the task-specific, context-aware natural language understanding (NLU) methodology implemented in MIA-PROM.

cient linguistic context is available, we observed substantially higher WER for short utterances consisting of only one or two words (e.g., German words corresponding to "yes", "no", "bad", "sometimes", etc.). Even when augmenting the ASR input with the full question text and additional explanatory context, the resulting WER for such brief utterances remained unsatisfactory.

Our second and final implementation approach employs a German speech recognition model (*vosk-model-de-0.21*) from the Vosk toolkit[4] developed by Alpha Cephei. This system operates on an incoming audio stream, continuously indicating the presence of speech and providing incremental updates of the recognized lexical units. For our application scenario, the substantially higher robustness in recognizing short utterances outweighs the limitations that Vosk returns all transcriptions exclusively in lowercase and typically renders numerical expressions as words. Furthermore, in contrast to Whisper, which generally requires a GPU to achieve (near) real-time transcription, Vosk has very good performance using only a high-performance CPU.

Due to the distributed architecture of the MIA-PROM system, the microphone and loudspeaker (tablet or avatar) are hosted on different devices. Consequently, it was necessary to implement an automatic muting and unmuting mechanism for the microphone whenever the system is speaking. Achieving precise temporal coordination of this functionality poses a challenge because of latency in the MQTT-based messaging. Moreover, using this approach, user barge-in during system output cannot be supported.

Natural Language Understanding Our natural language understanding (NLU) components are trained on utterances that were both expert-authored and empirically collected in user studies [3]. For each response scale, as well as for auxiliary tasks such as system-settings control and meta-communication, we train a dedicated NLU model. Each model receives LaBSE embeddings as input to a two-layer feed-forward neural network. Additional details and evaluation results on this *Many NLU* architecture are provided in [3].

Response Generation The MIA-PROM response generation component employs a straightforward and robust method based on predefined templates to produce textual output for both spoken and graphical user interface (GUI) modalities. In specific interaction scenarios, such as requests for repetition of the most recent system utterance, a set of alternative templates has been defined to increase linguistic variability and thereby mitigate perceived monotony in system responses.

Text-to-Speech Synthesis For deployment in the field experiments, we employed high-fidelity synthetic voices provided by Microsoft Azure. These voices are not available for offline use and therefore require a persistent Internet connection. However, the response latency of the

[4] `https://github.com/alphacep/vosk-api`

text-to-speech (TTS) service can be on the order of several seconds; in some instances, we observed delays of 30 seconds or longer, depending on the local capacity and quality of the mobile broadband connection.

To mitigate network usage and reduce reliance on real-time requests, a caching mechanism has been implemented. Comprehensive pre-generation of all possible system audio outputs is not feasible, as predicting the full range of triggering inputs is highly complex. Moreover, such pre-computation would need to encompass all variants of a single utterance across the full combinatorial space of loudness levels and speech-rate settings, leading to prohibitive storage and computational requirements.

3.2 Avatars and Questionnaire App

In addition to serving as representations of MIA, the tablet and avatars functioned as the output channels for the system's audio (speech) and provided visual feedback indicating the current state of the microphone.

Virtual Avatar The virtual agent (VA) is implemented as an iPadOS application executed on an iPad Air (11-inch). The *Unity* game engine serves as the primary development framework. The VA, represented by a female avatar with a warm and competent voice and appearance, was created using Daz Studio[5]. Lip movements are synchronized with the speech output via the SALSA LipSync Suite[6].

Physical Avatar The hardware and software development kits for the PA are provided by Furhat Robotics[7]. The Furhat robotic head was integrated into our system by implementing a custom skill, which received text strings to be synthesized via the MQTT protocol. Lip synchronization is handled natively by the Furhat platform. The same Azure TTS voice as that used for the PA and the tablet interface was employed for the Furhat. In addition to lip synchronization, the Furhat maintains eye contact with the user throughout the interaction. User detection and tracking are likewise provided by the Furhat platform.

Questionnaire App The tablet serves as the graphical user interface (GUI) for user interaction with the questionnaire and for the configuration of system settings. In addition, it manages the state of the questionnaire, including the selection and sequencing of questions to be presented, the corresponding response options, and the recording of user-selected answers.

The questionnaire is modeled according to the FHIR (Fast Healthcare Interoperability Resources) standard in order to facilitate seamless data exchange with other health-related information systems, such as hospital information systems (HIS).

The application is developed in C# and XAML using Microsoft's .NET Multi-platform App UI (MAUI) framework[8]. This technological stack enables cross-compilation of the application for Android and iOS/iPadOS platforms. For the field deployment, the application was executed exclusively on iPad devices. Consistent with the remaining system components, the application employs the MQTT protocol for message-based communication with the other modules.

3.3 Event Logging

The logging component is implemented in .NET C# and is configured to subscribe to all MQTT topics in order to record all system events, excluding recorded or generated audio files, into a SQL database. This persistent storage enables subsequent quantitative and qualitative analyzes

[5] `https://www.daz3d.com`

[6] `https://assetstore.unity.com/packages/tools/animation/salsa-lipsync-suite-148442`

[7] `https://www.furhatrobotics.com`

[8] `https://dotnet.microsoft.com/en-us/apps/maui`

of user–system interactions and facilitates systematic debugging throughout the development process.

4 Discussion and Conclusion

Our implementation was successfully deployed in two field trials, and the corresponding results are currently undergoing publication. The selected system architecture, in particular the design decision to employ MQTT as the communication protocol between heterogeneously implemented modules, proved to be effective overall. However, latency in the MQTT-based message exchange posed a challenge for achieving timely transitions between system speaking and listening modes during spoken interactions.

Moreover, the employed Vosk model for automatic speech recognition (ASR) exhibited adequate overall performance; however, it remains sensitive to speaker-specific characteristics (e.g., accents, disfluencies and pauses), as well as to background noise, side conversations, and systematic recognition errors (hallucinations), which continue to pose substantial challenges.

The limited availability of broadband internet in the clinics was not foreseen. Using clinical partners' WiFi was impossible for technical and organizational reasons. Stable mobile broadband was also difficult to obtain, even in the capital region and across three network providers.

ACKNOWLEDGMENT The described work has been funded by the *Federal Ministry of Research, Technology and Space (BMFTR)*, Germany, under grant no. 16SV9018 for the joint research project *MIA-PROM* in the research program *Interactive Technologies for health and Quality of Life*.

Special thanks to our project partners. Acalta GmbH developed the tablet app (including FHIR), the MQTT broker, and MQTT logging. dexter health GmbH provided GDPR-compliant access to high-quality Microsoft Azure TTS and implemented the skill integrating the Furhat robotic head into MIA-PROM. Charité – Universitätsmedizin Berlin and Hochschule München enabled field evaluation of the system at ZAR Berlin and Seehof Kliniken Teltow.

References

[1] HILLMANN, S., N. ASHRAFI, P. GRAF, P. L. HARNISCH, E. JANSEN, M. MARQUARDT, and J.-N. VOIGT-ANTONS: *Multimodal Interactive Assistance for the Digital Collection of Patient-Reported Outcome Measures*. In *The Digitalization of Healthcare for Older Adults*, pp. 99–108. Berlin Universities Publishing, Berlin, 2024. URL `https://doi.org/10.14279/depositonce-20351`.

[2] HILLMANN, S. and P. L. HARNISCH: *Multimodale Interaktive Assistenz zur Erhebung von Patient-Reported Outcome Measures*. In *Sprachassistenten - Anwendungen, Implikationen, Entwicklungen*, pp. 19–21. Regensburg, 2024.

[3] HARNISCH, P. L. and S. HILLMANN: *Empirical Evaluation of ASR and NLU in a Multimodal Dialogue System for Survey Answering*. In *Proc. 35. ESSV*, vol. 35, pp. 211–218. TUDPress, Regensburg, 2024. doi:10.35096/OTHR/PUB-7100.

[4] ASHRAFI, N., V. NEUHAUS, F. VONA, N. L. PEPERKORN, Y. SHIBAN, and J.-N. VOIGT-ANTONS: *Effect of external characteristics of a virtual human being during the use of a computer-assisted therapy tool*. In *Proc. HCII 2024*, pp. 3–21. Springer Nature Switzerland, 2024.