Robust Dialog Design for Emergency Situations with AI-Based Voice Assistants

David Feilacher[1], Sabine Lehner[1], Mario Heller[1], Bernhard Girsule[1] and Andreas Jakl[1]

[1]*University of Applied Sciences St. Pölten*

Abstract
With rising life expectancy and seniors seeking to live independently for as long as possible, automated emergency detection and communication will play a vital role in ensuring quick emergency responses in the future. Voice assistants enable a quick and hands-free reaction by the users and can also take their feedback into account. This paper presents the methods employed by the Smart Companion project to improve the dialogue and technical structures of a voice assistant system for emergency situations. We propose a dialogue structure, guided by a user centered design process, that issues concise and closed-ended questions while being able to process a wide range of possible user responses. This structure can handle any verbal user input, including non-responses and unintelligible utterances, through feedback and fallback mechanisms. To avoid technical errors or unresponsive users prematurely terminating an emergency call, conversations can be restarted at any point in the dialogue structure, by working in tandem with the other Smart Companion components. This also enables the emergency system to proactively ascertain if the user needs help. The technical implementation provides an easy dialogue configuration, adaptation, and expansion, with a custom state machine design supervising the dialogue flow.

Keywords
Voice Assistants, Human Machine Communication, AI, Dialogue Design, State Machine Design

1. Introduction

Communicating with people in distress can be difficult, even for experienced emergency personnel [1]. To automate this process with a Voice Assistant (VA), the dialogue design and implementation have to be technically robust, unambiguous and easy to follow for the user. This is important not only for successful emergency notifications, but also for preventing false alarms and reducing the costs for the public health system. The Smart Companion project [2] realizes an automated emergency assistant by combining a vacuum cleaning robot with a VA. It detects persons lying on the ground and interacts with them verbally, ascertaining whether they need help. A person in need can also invoke the VA to notify the emergency service proactively. The functionality is expanded by a variety of interactions such as robot control and a notification of trip hazards for preventing falls. This paper presents the development process and implementation of the Smart Companion's VA system, its dialogue structure, requirements, and technically robust implementation as an emergency system.

Section 2 first compares other personal emergency response systems with VAs, then shows how a current VA is typically configured and the problems emerging from it. Section 3 shows the development methods of dialogue requirements based on literature research, which are then improved upon by testing a basic VA configuration with potential users (user-centered design process). This is then translated to technical requirements as a guide for implementation. The final technical implementation is thereby a direct result of the user requirements. Additional techniques ensure unambiguous communication via feedback and fallback mechanisms and prevent technical errors from stopping a genuine emergency notification. The implemented dialogue configuration is modular, adaptable, and scalable. Section 4 presents the final abstract dialogue structure and its technical implementation, fulfilling all derived requirements. Section 5 gives a final overview of the results and an outlook for future improvements.

6th International Conference on Creative/Media/Technologies 2024, St. Pölten University of Applied Sciences, Austria
david.feilacher@fhstp.ac.at (D. Feilacher); sabine.lehner@fhstp.ac.at (S. Lehner); mario.heller@fhstp.ac.at (M. Heller); bernhard.girsule@fhstp.ac.at (B. Girsule); andreas.jakl@fhstp.ac.at (A. Jakl)
0009-0003-2390-0191 (D. Feilacher); 0000-0002-2953-4183 (S. Lehner); 0000-0003-2750-5634 (M. Heller); 0000-0002-2953-4183 (A. Jakl)

2. State of the Art

2.1. PERS and fall detection

Personal Emergency Response Systems (PERS) are devices or systems designed to enable individuals, especially seniors, to call for help in an emergency. They can be distinguished based on how they are activated and the types of sensors they use to detect emergencies [3]. Systems with manual activation require users to press a button on a wearable device, such as a pendant or wristband, to call for help. However, users can forget to carry the device or don't wear it on purpose [4]. Voice-activated systems allow users to summon assistance through verbal commands, they are however usually immobile and hard to interact with through doors and other obstructions. Passive monitoring systems continuously observe the user's behavior or environment and trigger alerts when abnormal patterns are detected. Being constantly monitored with a camera has obvious privacy issues if the data is analyzed or logged in the cloud. Furthermore, for complete coverage, each room would have to be monitored, making the system expensive to install.

Fall detection systems can use a long list of sensor technologies for detecting possible falls ([5] gives an overview). Accelerometers, radar, ultrasonic, pressure sensors and motion sensors can identify sudden impacts or suspicious movements which might indicate an emergency. GPS-enabled systems provide real-time location tracking, ensuring users can be located quickly.

Combining multiple activation methods and sensor technologies enhances reliability and reaction time. The Smart Companion project uses a combination of image analysis and voice interaction. The system can be activated via voice commands, but also detects persons lying on the ground itself and can call for help if the person is non-responsive. Since the robot is mobile, it can examine every room in search runs and returns to its docking station once finished. This provides complete apartment coverage without constant surveillance. The user can choose frequency of search runs. This way, a balance between privacy concerns and reaction time can be found.

2.2. Voice Assistant Systems

Most current VA systems allow for the creation of third-party dialogue designs [6, 7], commonly called skills. Creators can define a name to invoke the VA and its intents with respective utterances. Intents group utterances according to their intention or meaning, for example the utterances "Yes" and "Yes, please" can be grouped to the "Yes" intent. After the intent is identified, the skill code decides how to respond to the user. This basic principle is depicted in Figure 1. The VA of the Smart Companion project is implemented with the Amazon Alexa Skills Kit (ASK) [8].

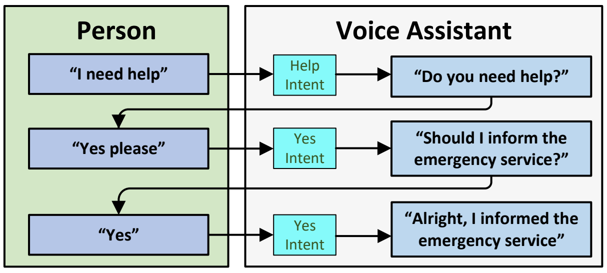


Figure 1: Example conversation with a VA in an emergency situation

Figure 2 shows in more detail how the interaction between the user and VA is processed. Starting from the person symbol on the left, verbal user statements are transformed into text with Automatic Speech Recognition (ASR) then they are matched to an intent. For this purpose, the ASK skill designer provides utterances for each intent, which an AI-supported Natural Language Understanding (NLU) uses for matching the users responses to intents. If an intent is matched successfully, a respective user-defined function (intent function) is executed and derives a response in text form. This text is then translated via text-to-speech and then played back for the user. The NLU provides a certainty value for the intent match. If the match score is too low, the fallback intent code will be executed; if no response is detected, a timeout function is called.

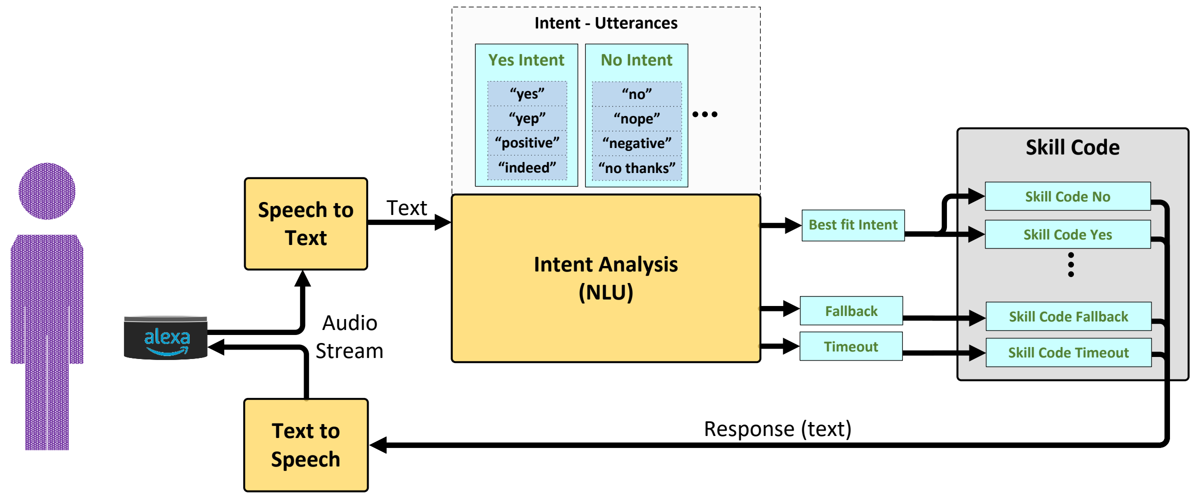


Figure 2: Overview of a VA system's basic function, starting from the user on the left, the verbal user input is analyzed and a response generated depending on the intent functions

To activate a specific function, the user first has to utter the wake word (activation phrase for the VA), the skill name, and one of the respective intent utterances (e.g. "Alexa, tell Smart Companion to start cleaning"). After the interaction has been started, the skill will continuously respond to the user according to the intent code until either the user does not respond anymore or the skill program ends the session. With these intent functions, the programmer can create a dialogue and decision structure to organize the VA's answers and actions. An Alexa skill cannot start its program execution by itself; the Smart Companion circumvents this by mounting a small speaker to the VA's microphone and using an external control unit to play back a voice recording that triggers skill execution.

The functionally distributed way in which the user-defined code is executed presents challenges for creating a structured and scalable dialogue. As Figure 3 shows, a dialogue design can be represented by a flow chart where the conversation manifests by following a certain path, depending on the users' answer. During execution, the skill only decides whether to switch to the next Section and then derives the appropriate response. To track its current position within the dialogue flow and assess potential subsequent intents, each intent function needs to recall previous interactions. It then has to determine whether its activation can advance the flow and how it should do so. If a certain intent is not expected at a certain point in the dialogue flow, its function code has to be aware of this fact and take actions that may have nothing to do with its actual intent.

During the same conversation each intent can also be called multiple times with multiple different purposes. Therefore, the dialogue flow from Figure 3 (top) has to be split between multiple intent functions, containing code for eventualities that may have nothing to do with this intent. This makes the code cumbersome and redundant but also increases the chances of introducing errors. If one part of the dialogue design is changed, its code has to be moved or adapted in multiple places. Errors in one intent function can affect the larger dialogue flow.

This complexity is compounded by reacting to timeouts, technical error messages, and the addition of higher-level decisions such as emergency levels and input prioritization where the behavior of the system is changed depending on previous interactions (Figure 3 bottom). As mentioned, an Alexa skill can not reactivate itself in the event a conversation ends unintentionally, e.g. through timeouts or technical errors, which is a major problem for the reliability of an emergency system.

To reduce the redundancy and complexity, the dialogue flow can be controlled centrally as the decision process is always the same:

- Check the incoming intent
- If intent is expected, switch to next dialogue (according to dialogue design)
- If not, react accordingly (also for timeouts, other errors and modes)

However, with the intent-bound nature of the VA's program execution, a centralized control is made difficult. The VA only executes the specific intent function matching the user's utterance. Section 4 presents a solution to the technical issues presented here.

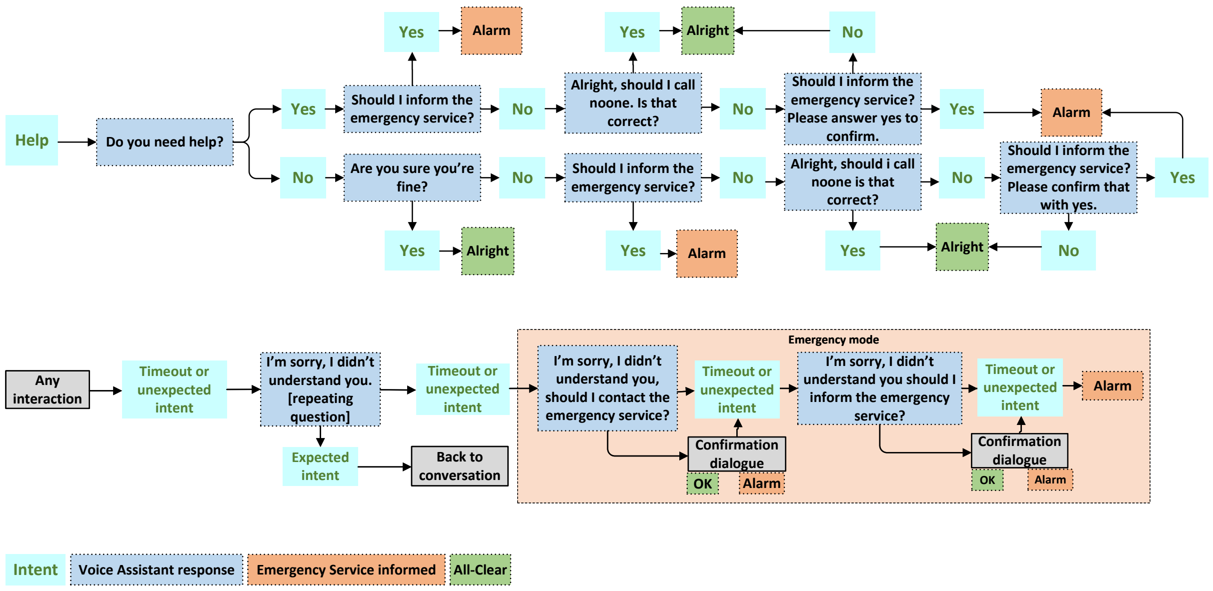

Help
Do you need help?
Yes
No
Should I inform the emergency service?
Are you sure you're fine?
Alarm
Alright
Alright, should I call noone. Is that correct?
Should I inform the emergency service?
Should I inform the emergency service? Please answer yes to confirm.
Alright, should i call noone is that correct?
Should I inform the emergency service? Please confirm that with yes.
Any interaction
Timeout or unexpected intent
I'm sorry, I didn't understand you. [repeating question]
Expected intent
Back to conversation
Emergency mode
I'm sorry, I didn't understand you, should I contact the emergency service?
I'm sorry, I didn't understand you should I inform the emergency service?
Confirmation dialogue
OK
Alarm
Intent
Voice Assistant response
Emergency Service informed
All-Clear

Figure 3: *Top:* Split of dialogue flow between intent invocations; *Bottom:* change in behavior with additional dialogue escalation modes for emergencies

3. Methods

This section explains the development process of the VA's dialogue, starting with the creation of Dialogue Requirements (DRs) (Section 3.1) which are basic rules for the dialogue design. They are then used as guidelines for creating a dialogue structure and are translated into technical requirements (Section 3.2.1), leading to the implementation principles (Section 3.2.2).

3.1. Dialogue Requirements

Part of the DRs were found in a literature review on human-machine interaction, with a focus on elderly users and voice assistants done by Lehner [9] as part of the conceptual phase of the Smart Companion project. Section 3.1.1 briefly summarizes her findings and derives the respective DRs. Using these results, a basic version of a Alexa Skill was created and with it the DRs were refined and expanded by a user study conducted by Lehner [9] (described in Section 3.1.2).

3.1.1. Theoretical Background

Dialogue requirements DR1, DR3, DR4, DR5, DR6, DR7, DR8 and DR11 are based on sociolinguistic and human-machine interface literature on voice assistant interaction in assistive contexts, particularly for older adults [10, 11, 12, 13].

DR1: Kopp et al. [10] argue that socially cooperative dialogue can improve user acceptance but warn against excessive anthropomorphization, as it may mislead users about the VA's actual capabilities. Therefore, DR1 was formulated to ensure a friendly but clearly machine-like VA persona.

DR2: Trajkova and Martin Hammond [11] and other studies cited in Lehner [9] emphasize that privacy concerns are a major barrier for older adults adopting voice assistants. To address this, DR2 was instruduced to ensure minimal data collection and to clearly communicate privacy boundaries.

DR3: Cyra and Pitsch [12] highlight that long, open-ended user responses often caused breakdowns in dialogue. To prevent this, DR3 was introduced, requiring closed-ended and simple prompts. A clear yes/no question format ensures better comprehension and usability.

DR4: Lopatovska and Oropeza [13] report that technical problems like microphone distance or internet instability can disrupt communication and lead to user frustration. They emphasize the importance of polite re-prompts to recover the dialogue. Based on this, DR4 was defined to include such clarification strategies. Re-prompts should be polite and repeat the core question to reorient the user.

DR5: In emergency situations, a clear and assertive response strategy is necessary. [10] and [12] both highlight that vague or indirect dialogue can slow down user decision-making. As a result, DR5 was introduced to ensure the VA uses direct and goal-oriented language under stress.

DR6: According to [11], older users value control and autonomy in interactions. To reflect this, DR6 was added, allowing users to abort the dialogue at any point.

DR7: [11] also show that users frequently employ informal or dialectal language, which standard VAs struggle to interpret. Thus, DR7 was formulated to encourage inclusive input design, including dialect variants.

DR8: [13] further emphasizes that repeated misunderstandings require a shift in system behavior. Escalating tone and urgency can help maintain dialogue flow and highlight the importance of the situation. Therefore, DR8 was established to define escalation strategies.

DR11: Finally, [11] stress that system accessibility improves when users can adjust voice parameters such as speed, volume, and pauses. This led to the inclusion of DR11, ensuring configurability of speech output.

These findings formed the basis for further refinement through user testing in Section 3.1.2.

3.1.2. User Centered Refinement

To include the feedback of potential users, first a basic version of an Alexa skill was created satisfying the requirements from Section 3.1.1. Then a study was conducted to gauge their reaction and use it for the refinement of the DRs. Six senior citizens (P1-P6) participated in task-based interactions and accompanying interviews. For most of them, this was the first time interacting with Alexa. The following is a summarization of the study's insights that lead to the validation, adaptation and expansion of the established dialogue requirements from Section 3.1.1.

DR2: During the interviews, several participants expressed explicit privacy concerns, particularly regarding continuous listening and data handling (see Lehner [9], participants P1, P3, P5). While emergency use cases were viewed more tolerantly, this generally conrifmed DR2.

DR4/DR5: In emergency scenarios, some participants responded with ambiguous or contradictory statements which the VA can not correctly interpret. For example, P4 initially asked for help but later said: „Naja, aber so schlimm ist es vielleicht doch nicht…“ To prevent confusion and ensure the right action is taken, DR5 was adjusted to give clear options during the conversation (e.g. if they want to alert the emergency service. or not). This also confirmed the importance of DR4 stating which answeres are possible ("yes" /"no")

DR6: Multiple participants stated they wanted to end the interaction at will, especially if the system was not performing as expected. As P2 put it: „Ich will das dann einfach beenden können – wie beim Telefon auflegen.“ This confirmed the importance of DR6 and the resulting "GoHome" command.

DR7: Dialectal expressions like „jo“ or „na“ were frequently used and not always understood by the system. While DR7 had already been defined, these findings led to a further addition of utterances for intent activation. Another addition was adding utterances with different word order. Both insights led to utterance expansions for example for the "GoHome" - Intent: original:"fahr nach hause", different order: "nach hause fahren" and dialectal: "schleich dich".

DR9: In situations where users remained silent or gave unclear responses, project members identified the risk of missing critical emergencies. To handle such scenarios, DR9 was introduced: the system should proactively re-engage the user to determine if assistance is needed.

DR10: Finally, outside of the user study during internal tests and exploratory use by researches, participants repeatedly gave the same input to test system consistency. This behavior revealed the risk of circular dialogues or stagnation. To counter this, DR10 was formulated to ensure a loop-free, clearly progressing dialogue structure.

3.1.3. Combined Dialogue Requirements

Through the findings of Section 3.1.1 and the observations during the study in Section 3.1.2, the dialogue requirements summarized in Table 1 were derived. They will be translated to technical requirements in Section 3.2.

3.2. Technical Requirements

This Section describes the technical requirements (TRs) derived from the DRs of Section 3.1 and additional methods to improve technical robustness. The resulting TRs are summarized in Table 1.

3.2.1. Derivation of Technical Requirements

Dialogue requirements DR1, DR2, DR3, DR5, DR10 and DR11 do not depend on the technical implementation but on the abstract dialogue configuration. DR4 was partially already implemented with the ASK. If the user response is not understood or the user does not respond at all (timeout) a response can already be configured (“re-prompt”) in ASK. Additionally TR1 is introduced for intents that are correctly identified by the Alexa NLU, but are not expected at a certain part in the dialogue flow. For this an “unexpected intent re-prompt” is created that repeats the question, similarly to the “re-prompt”. DR6 suggest the system should be able to prioritize certain user input, leading to TR2. For example the abort intent “GoHome” has the highest priority. When it is detected the dialogue is ended and in the wider Smart Companion context, the robot drives home. DR7 suggests to add multiple utterances to each intent, this is not only good for dialect understanding but also helps the NLU to learn identifying intents better. These utterances are set in the ASK itself. Additionally, TR3 states

No.	Dialogue Requirement
DR1	Use friendly, calm, and familiar language, but limit overly anthropomorphized responses to avoid confusion about the VA's actual capabilities.
DR2	Avoid asking for or recording unnecessary information and assure users of the VA's limited data collection for privacy comfort.
DR3	Avoid complex question structures or lengthy interactions. The VA should follow a "yes / no" or short, directive-based response format.
DR4	In case of misunderstanding, repeat or confirm actions in a non-intrusive way to ensure clarity without frustrating the user. Provide gentle prompts or corrections (e.g., "I'm sorry, I didn't understand you. Please only state 'yes' or 'no'.") to guide users to give a concise answer.
DR5	Adopt a simplified and assertive response style in emergencies. It should give clear options without requiring users to repeat commands if they want to alert the emergency service.
DR6	The user should always be able to stop the current conversation with an abort utterance, e.g., "Go home".
DR7	Accommodate variations in dialects, incomplete sentences, or informal language, reducing the need for exact phrasing.
DR8	In case of a possible emergency situation (e.g., long periods of silence or unexpected answers), shift to more alarming and demanding VA statements.
DR9	In case of a possible emergency situation, the VA should proactively try to interact with the user to see if help is needed.
DR10	Create and verify a loop free dialogue structure.
DR11	Adjust voice speed, volume, and pauses.

Table 1: Table of dialogue requirements found through via literature review and the user centered design process

adding more possible intents for dialogue flow control (see Figure 4). This means, even though only "yes" and "no" intents should be incited by the VA, it should still understand multiple other intents that would fit the context. For example, if the user answers wether they need help or not, they can answer yes but also repeat the cry for help ("Help" intent) or say they are hurt ("ImNotOk" intent). DR8 suggests different modes (normal/emergency) changing the system behaviour with repeated timeouts or unidentified utterances, translated to TR4.

DR9 requires an external control unit (on the Smart Companion robot) so the VA can not cut off an emergency dialogue due to simple technical errors or misunderstandings. This is prevented by TR5 necessitating detection of such problems within the Smart Companion system and TR6 for restarting the VA dialogue at any point. For emergency systems traceability is generally required so TR12 (logging) is added. This also enables analyzing the user interactions with the VA for further improvements. TR5, TR6 and TR12 lead to TR7, the implementation of a bi-directional communication interface, to interact with the external control unit. This communication not only enables error feedback, conversation restarting and logging, but also activation of the actual emergency call, and further information exchange (e.g. for adapting responses).

TR8 requires a high certainty value for intent identification, set for the NLU, thereby reducing the probability of misunderstandings and instead repeating the dialogue for clarity. TR9 demands the ability for easy configuration of the dialogue structure based on an abstract flow chart (e.g. Figure 4). The configuration should be modular, adaptable and scalable (TR11) to quickly integrate feedback of a later pilot study (adding trip hazard notification, etc.). This should be done without changing the program code itself, so the introduction of code errors can be avoided (TR10). All technical requirements are summarized in Table 2.

3.2.2. Technical Implementation Prelude

This Section shows how the technical implementation principles are derived from the TRs of Table 2 as a prelude to Section 4.2.

No.	Technical Requirement
TR1	Re-prompt for unexpected intents.
TR2	Signal prioritization for immediate exit from a conversation at any point.
TR3	Configuration option for multiple intents as user response.
TR4	Configurable behavior change with emergency modes.
TR5	External control unit to detect interrupted emergency call and information exchange (in a secure channel).
TR6	Ability to start/restart the conversation at any point in a defined dialogue structure (controlled by external control unit).
TR7	Bi-directional communication interface for TR5 and TR6 and additional information exchange with external control unit.
TR8	Setting a high certainty value for identifying intents.
TR9	Easy dialogue structure configuration based on abstract flow charts.
TR10	Reconfiguration of dialogue structure without introducing programming errors.
TR11	Modular, adaptable and scalable configuration.
TR12	Logging for traceability of decisions.

Table 2: Table of technical requirements to satisfy dialogue requirements and technical robustness

Technical requirements TR1 and TR3 necessitate a storage for all parameters of each dialogue section. This includes the prompt, re-prompt, unexpected-intent prompt and a list of intents for switching to the next section of the dialogue (see Figure 4 for reference). Since additional code for communication with the external Smart Companion controller is also necessary (TR7) and depends on the current dialogue section, this will be a combined data entity that contains parameters as well as program code for communication. Furthermore, the data entity has to contain an emergency mode parameter for TR4 and the overlying control must implement the respective behavior changes.

TR2 suggests that the overlying control must be able to react to certain inputs in a generally set manner, independent of the individual dialogue sections (centrally controlled). For example multiple non-responses or the “GoHome” intent must always be acted upon. This and TR2, TR4, TR6 and TR11 suggest a centralized control that loads these data entities according to the user input, configuration and due to communication from the external control unit (TR7). Once loaded, the control program responds to the user according to the configured prompt and again, waits for the next user input. This repeats until the conversation is finished either by calling the emergency services, activating a number of robot tasks (e.g. going home).

According to TR6, the overlying control must be able to start a conversation at any point, so loading of a data entity has to set all required parameters and not be influenced by any dialogue from before. Therefore the ASK skill does not require a long term memory (e.g., a database) as the external controller of the Smart Companion handles the overarching system logic, e.g. the number of times a timeout was detected for escalating the emergency mode. This reduces the complexity of the ASK skill.

Requirements TR9, TR10 and TR11 suggest a dialogue structure configuration that reflects the data entity mentioned above, but is detached from the program code itself. Ideally a graphical user interface would be implemented to simply creating a graph like shown in Figure 4, however making this work reliably would introduce unnecessary complexity in the overall project. A text based configuration is a better option in this case. If the configuration is based on an program-external text file, once the control program code is finished and validated, changes in the configuration also can not introduce code errors. Unexpected behavior could still be introduced by an inconsistent configuration. However, by adding validation functionality (configuration checker), the likelihood of this occurring can be greatly reduced.

To summarize, the system should be centrally controlled loading certain dialogue sections from a configuration. Once loaded, certain actions are executed (e.g. sending a response and exchanging information with the external control unit), then the system remains in this state until one of its exit conditions is met (e.g., new user input). It can then switch to the next dialogue section. This behaviour can be implemented with a finite state machine. A finite state machine [14] is a very robust control program based on discrete states, each representing a specific condition or mode in which the program resides. These states can execute a subprogram and then wait for a defined condition to exit to the next state. As shown in [15], state machines are well known models suited to describing reactive behaviors. Section 4.2 will describe the implementation of the Smart Companion state machine in more detail.

4. Results

This section will explain how the requirements from Section 3.1 and 3.2 are satisfied and implemented. First, an example for the abstract dialogue structure is presented in Section 4.1, showing how to follow the dialogue requirements. In Section 4.2 the technical solution is derived from the technical requirements and an example dialogue is implemented with it, showcasing its validity.

4.1. Dialogue Structure

Figure 4 shows a small excerpt of the Smart Companion dialogue design. The dialogue structure is configured by defining conversation states (gray boxes, see Section 3.2) and linking them, which is represented by abstract connectors. The project has over 40 such conversation states, so only a small portion is presented here. It starts on the left with the “Help” intent being invoked by the user, showing the VA prompts (responses) and where detected intents lead the user in the dialogue flow.

The VA responses are friendly but direct, short, and concentrate only on ascertaining the need for help. Every question can be answered with the “yes” and “no” intent (Requirements DR1, DR2 and DR3 of Table 1). As suggested by DR4 and DR5, user input is initially confirmed. However, if users contradict themselves (e.g., first request help but then invalidate their statement), the responses become more assertive and suggest a possible yes/no answer. If the timeout/unexpected intents are detected once, the VA responds with the re-prompt/unexpected intent prompt respectively. For simplicity, the re-prompts and unexpected intent prompts are not explicitly shown in Figure 4 but they can be seen in Figure 8 in Section 4.2. These prompts state that the user’s answer was not understood and restate the previous response (prompt). If the timeout/unexpected intents are detected twice in a row, they lead to other sections of the dialogue flow not depicted in this figure, escalating the emergency mode. If this repeats another time it suggests the user may be in distress, so the system will notify the emergency service (as suggested by DR8 and DR9).

To satisfy DR6, the “GoHome” intent directly leads to the green exit message, enabling the user to exit the conversation at any point without alerting the emergency service. Due to DR7 a number of optional intents and respective utterances where provided to cover a wider range of possible user responses. For example when the VA asks “Do you need help?”, the user could respond with a general “yes” (or similar utterance) but also repeat their plea for help with another “Help” intent or say that they are not fine by invoking the “ImNotOk” intent (contains utterances proclaiming the person is not well). Overall, there are 21 intents with 398 utterances, an average of around 19 utterances per intent. The utterance variations include Austrian dialect (“nein”, “na”), completely different phrasings with a similar meaning (“I need help”, “I am hurt”), the manner in which the VA is addressed (“Call somebody”, “That it should call somebody”) and polite responses (“Yes please”, "Yes I feel fine and you?”). Even though nodes can have multiple connections to other nodes, no combination of user inputs can lead to an endless loop, satisfying DR10.

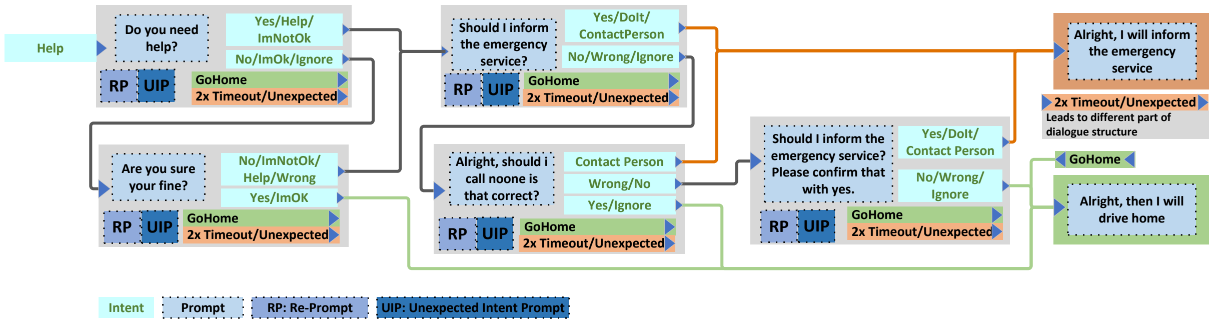

Help
Do you need help?
Yes/Help/ImNotOk
No/ImOk/Ignore
RP
UIP
GoHome
2x Timeout/Unexpected
Are you sure your fine?
No/ImNotOk/Help/Wrong
Yes/ImOK
Should I inform the emergency service?
Yes/DoIt/ContactPerson
No/Wrong/Ignore
Alright, should i call noone is that correct?
Contact Person
Wrong/No
Yes/Ignore
Should I inform the emergency service? Please confirm that with yes.
Yes/DoIt/Contact Person
No/Wrong/Ignore
Alright, I will inform the emergency service
2x Timeout/Unexpected
Leads to different part of dialogue structure
GoHome
Alright, then I will drive home
Intent
Prompt
RP: Re-Prompt
UIP: Unexpected Intent Prompt

Figure 4: Dialogue structure excerpt, satisfying the dialogue requirements from Table 1

4.2. Technical Implementation

This section describes how the developed dialogue structure from Section 4.1 is implemented with the methods suggested at the end of Section 3.2.

4.2.1. Centralization of code execution

To free the system of the dialogue splitting described in Section 2 and to satisfy the technical and dialogue requirements, each intent function call is treated merely as an input to the main control program. Since code is only executed when an intent is called, in practice this means every intent function executes the same function (main control program) and passes its name and parameters to it. The actual dialogue and decision code is thereby centralized in its function, although its invocation is technically still distributed. This principle is shown in Figure 5. The green arrows lead from the detected intents directly to the main control program (in this implementation a state machine), passing information about the calling intent.

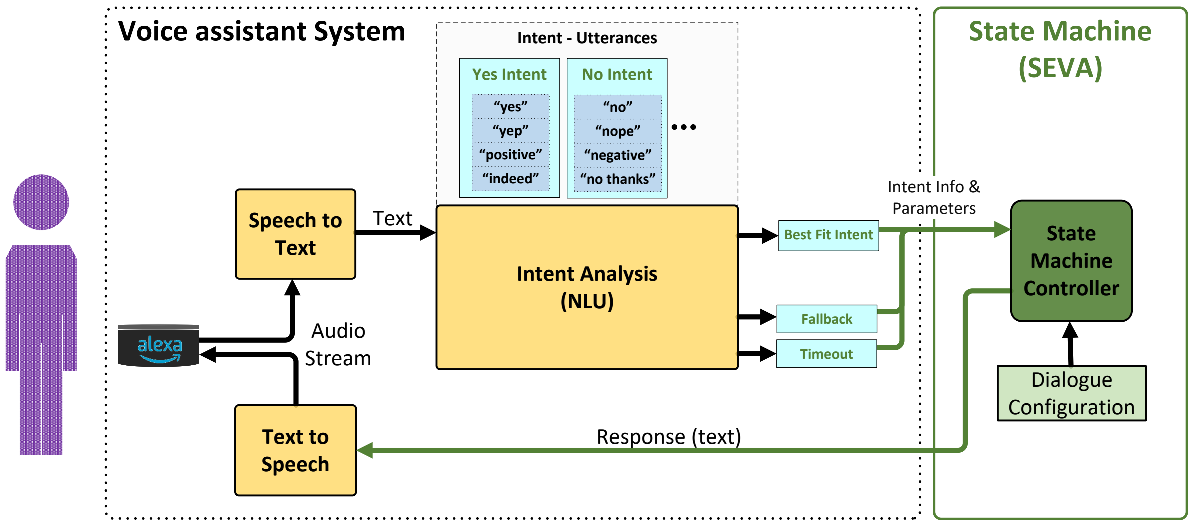
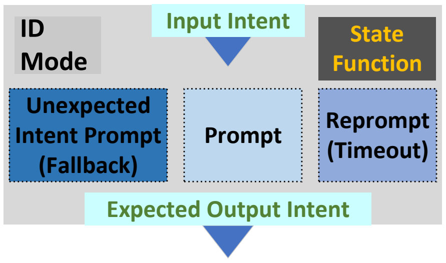


Figure 5: VA System with implemented State Machine for Emergency Voice Assistance (SEVA)

4.2.2. State Machine for Emergency Voice Assistance (SEVA)

As described in 3.2, the main control program is implemented with a state machine. In this work's technical implementation, a state represents an interaction between user and VA. Figure 6 (left) shows this state component concept.



```
{  ID: "Root_Help_E",
   Mode: "Emergency",
   Prompt: "Do you need help ?",
   Reprompt: "I did not understand you. Do you need help ?",
   UnexpectedIntentPrompt: "Im sorry I did not understand you
        . Do you need help ?",
   InputIntents: ["Help"],
   OutputIntents: ["Yes", "Help","ImNotOk","No","ImOk","
        Ignore"],
   StateCode: ["StopRobotFunction"] }
```

Figure 6: State component concept visualization and example state in JSON format

A functional schematic can be seen in Figure 7. The current state can be switched with a specified input intent, which would then respond with the specified prompt to the user and execute the state function. The state function is used for additional functionality such as the interaction with the wider Smart Companion System, which also controls aspects of the overall dialogue flow. This includes the retrieval of information for adapted responses, the emergency service invocation, connection testing and other technical and interaction details. This is done with a REST API interface because it is a common and easy way to exchange information over networks [16].

In case of an unexpected intent or a timeout (no input from the user is recognized), the re-prompt and unexpected intent prompt repeat a similar interaction. Two timeouts or unexpected intents in a row will switch the system into an escalated emergency mode repeating the questions in a more urgent and assertive manner. It is therefore not part of the state but of the general dialogue structure and control. After two timeouts, the ASK ends a conversation automatically. Therefore, the external control unit restarts the conversation in the escalated emergency mode. If once again no expected input is received, the system will alert the emergency service.

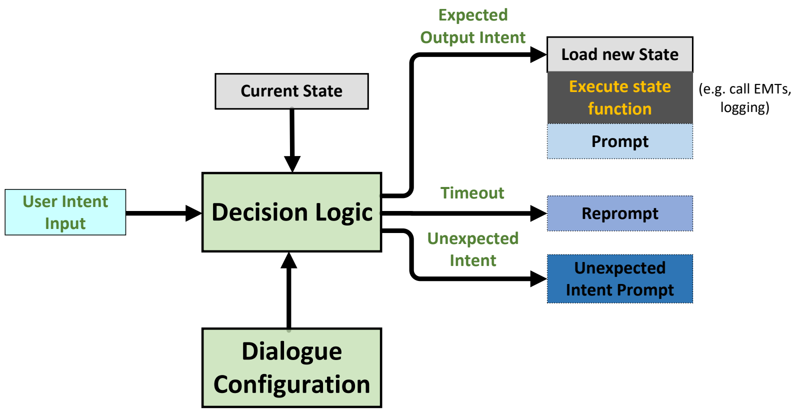


Figure 7: Basic function of SEVA State Machine, user input can lead the decision logic to switch states. Prompts are then sent to the user and the function code executed

4.2.3. JSON Format Configuration and Model Checker

The actual dialogue configuration is done via a JSON file [17] (Figure 6 right) where the settings represent the discussed parameters of a given state. To validate the configuration, a model checker looks for structural errors in the created dialogue flow with a given set of rules and will not allow the state machine to start until they are satisfied. Examples for such rules include: each connection must come from and lead to an existing state and an output intent can only be connected to one input intent. Since the configuration of the dialogue flow is done outside of the actual program and after the model is validated, it can't break the program flow or cause any errors during execution. This also enables an easy switch between model files and versions.

The state machine controller (Figure 5 right, green component) is the main control program and responsible for state switching, prioritization of events, error handling and additional technical and dialogue related decisions. It was designed specifically for the Smart Companion project to better satisfy the requirements defined in Section 3.1 and 3.2, its implementation specifics are however outside the scope of this paper. Figure 8 depicts the excerpt of the implementation of the dialogue flow from Figure 4 with the complete state definitions and their settings.

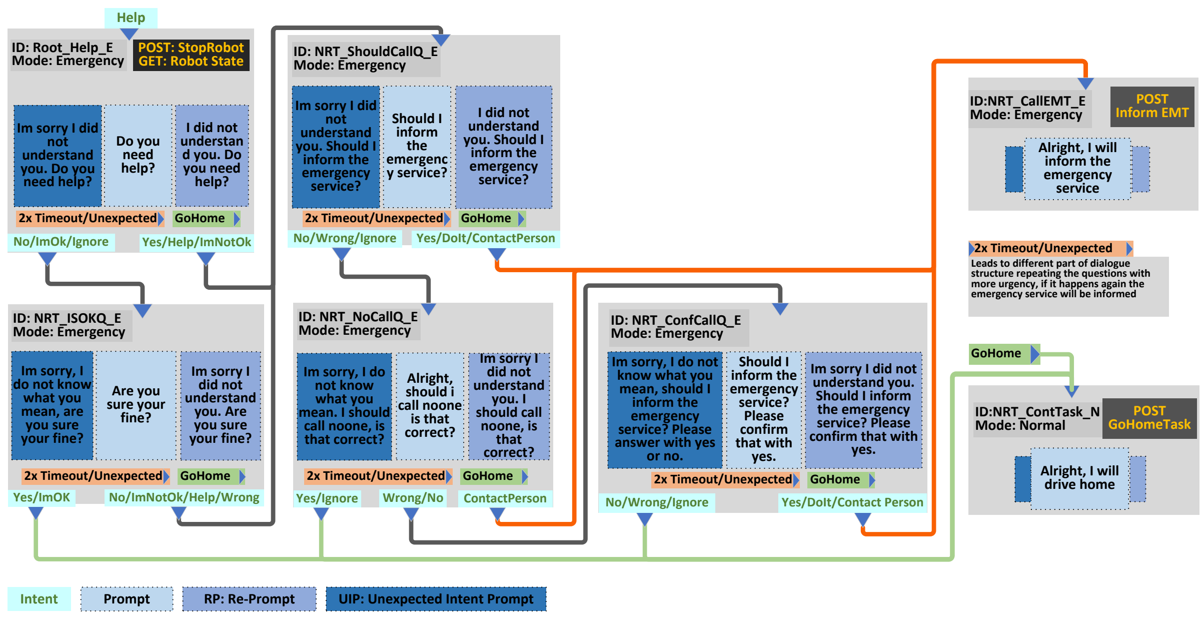

Help
ID: Root_Help_E
Mode: Emergency
POST: StopRobot
GET: Robot State
Im sorry I did not understand you. Do you need help?
Do you need help?
I did not understand you. Do you need help?
2x Timeout/Unexpected
GoHome
No/ImOk/Ignore
Yes/Help/ImNotOk
ID: NRT_ShouldCallQ_E
Mode: Emergency
Im sorry I did not understand you. Should I inform the emergency service?
Should I inform the emergency service?
I did not understand you. Should I inform the emergency service?
2x Timeout/Unexpected
GoHome
No/Wrong/Ignore
Yes/DoIt/ContactPerson
ID:NRT_CallEMT_E
Mode: Emergency
POST
Inform EMT
Alright, I will inform the emergency service
2x Timeout/Unexpected
Leads to different part of dialogue structure repeating the questions with more urgency, if it happens again the emergency service will be informed
GoHome
ID: NRT_ISOKQ_E
Mode: Emergency
Im sorry, I do not know what you mean, are you sure your fine?
Are you sure your fine?
Im sorry I did not understand you. Are you sure your fine?
2x Timeout/Unexpected
GoHome
Yes/ImOK
No/ImNotOk/Help/Wrong
ID: NRT_NoCallQ_E
Mode: Emergency
Im sorry, I do not know what you mean. I should call noone, is that correct?
Alright, should i call noone is that correct?
Im sorry I did not understand you. I should call noone, is that correct?
2x Timeout/Unexpected
GoHome
Yes/Ignore
Wrong/No
ContactPerson
ID: NRT_ConfCallQ_E
Mode: Emergency
Im sorry, I do not know what you mean, should I inform the emergency service? Please answer with yes or no.
Should I inform the emergency service? Please confirm that with yes.
Im sorry I did not understand you. Should I inform the emergency service? Please confirm that with yes.
2x Timeout/Unexpected
GoHome
No/Wrong/Ignore
Yes/DoIt/Contact Person
ID:NRT_ContTask_N
Mode: Normal
POST
GoHomeTask
Alright, I will drive home
Intent
Prompt
RP: Re-Prompt
UIP: Unexpected Intent Prompt

Figure 8: Excerpt of implemented dialogue design visualized

5. Summary and Conclusion

In this work we developed a emergency VA dialogue structure and technical implementation for older people, based on a user centered design process. The creation of requirements for individual dialogue sections and their combination were based on literature research validated by various user testings. These requirements where iteratively adapted and then transformed to an abstract dialogue structure and technical requirements necessary to implement it in the Alexa Skills Kit and working with the components of the Smart Companion project. Special attention was put on unambiguous VA responses enticing clear answers while being able to cope with a range of user responses. The abstract dialogue structure ensures a loop free and traceable dialogue flow. With its configuration directly mappable from a flowchart, it allows for seamless transition from a graphical dialogue design to the text based configuration. On the technical side, through centralization, generalized behaviour strategies could be implemented for perceived emergencies, as well as for a quick exit of the emergency dialogue. The state machine based solution ensures that the system is always in a defined state and switches to a new dialogue section only with an expected user input.

The system was tested during a later pilot study. Six test subjects (age 60+) participated each in a one-month home test. They where provided a Smart Companion system which was connected to the

ASB (Arbeiter Samariter Bund) home emergency system. Users gave the system an overall very positive assessment, some of them even requested more communication with the VA. Due to the modular and adaptable configuration of our system, their feedback was quickly integrated, and the dialogue design was updated after each test run without issues. During six runs of continuous one-month testing, no technical errors were recorded in the VA system. Impaired hearing was an issue among some of the test subjects, even with the VA at maximum volume. Impaired hearing was identified as an issue among some of the test subjects, even with the VA operating at maximum volume. As described by Acosta [18], specific warning sounds are more easily perceived by individuals with hearing impairments and could be integrated to enhance communication. Such auditory cues could supplement the implementation of DR8 (see Table 1) by helping to capture the user's attention in emergency situations where the system already employs more alarming and demanding voice statements. Additionally, they could be used to try and wake up sleeping users to assess responsiveness, or as part of fall prevention strategies—for example, to alert users when the robot is moving through the area.

The ability of large language models (LLMs) to generate dynamic and context-aware responses offers potential for enhancing user interaction in the Smart Companion VA. However, LLMs are known to be unreliable in decision-making and lack true situational understanding. Therefore, they are not suitable for handling entire conversations or making critical decisions, such as whether to contact emergency services. Instead, LLMs could be integrated into the reliable dialogue framework presented in this work. Specifically, they could be used to analyze user utterances and map them to predefined intents, improving flexibility in how the system interprets input and generates responses. This would help address limitations associated with using only closed-ended questions in emergency situations, as discussed in Section 3.1. In this setup, the LLM would extract meaning from user input and pass it to the existing state machine. If no confident intent match can be made, the system would fall back to a closed-ended clarification question. This approach ensures that control remains with the rule-based dialogue logic while leveraging the strengths of LLMs. Eder [19] has demonstrated this principle using the Smart Companion dialogue structure.

References

[1] J. Svennevig, On being heard in emergency calls. the development of hostility in a fatal emergency call, Journal of Pragmatics 44 (2012) 1393–1412. doi:10.1016/j.pragma.2012.06.001.

[2] St. Pölten University of Applied Sciences, Smart companion 2, Accessed: 2024-10-24. URL: https://research.fhstp.ac.at/projekte/smart-companion-2.

[3] P. Vallabh, R. Malekian, Fall detection monitoring systems: a comprehensive review, Journal of Ambient Intelligence and Humanized Computing 9 (2018) 1809–1833. URL: https://doi.org/10.1007/s12652-017-0592-3. doi:10.1007/s12652-017-0592-3.

[4] B. Heinbüchner, M. Hautzinger, C. Becker, K. Pfeiffer, Satisfaction and use of personal emergency response systems, Zeitschrift für Gerontologie und Geriatrie 43 (2010) 219–223. URL: https://doi.org/10.1007/s00391-010-0127-4. doi:10.1007/s00391-010-0127-4.

[5] P. Vallabh, R. Malekian, Fall detection monitoring systems: a comprehensive review, Journal of Ambient Intelligence and Humanized Computing 9 (2018) 1809–1833. URL: https://doi.org/10.1007/s12652-017-0592-3. doi:10.1007/s12652-017-0592-3.

[6] D. Coates, Voice Applications for Alexa and Google Assistant, Manning Publications, Shelter Island, NY, 2019. URL: https://www.manning.com/books/voice-applications-for-alexa-and-google-assistant.

[7] K. Jaber, M. Lafi, A. Alkhatib, A. AbedAlghafer, M. Abdul Jawad, A. Ahmad, Comparative study for virtual personal assistants (vpa) and state-of-the-art speech recognition technology, International Journal of Computational and Experimental Science and Engineering 10 (2024). doi:10.22399/ijcesen.383.

[8] A. Kumar, A. Gupta, J. Chan, S. Tucker, B. Hoffmeister, M. Dreyer, S. Peshterliev, A. Gandhe,

D. Filiminov, A. Rastrow, C. Monson, A. Kumar, Just ASK: Building an Architecture for Extensible Self-Service Spoken Language Understanding, 2018. URL: http://arxiv.org/abs/1711.00549. doi:10.48550/arXiv.1711.00549, arXiv:1711.00549.

[9] S. Lehner, The design and discursive construction of a 'speaking' vacuum cleaning robot for assistive purposes: Findings on communication ideologies from a current research and development project, AI-Linguistica. Linguistic Studies on AI-Generated Texts and Discourses 2 (2025). doi:10.62408/ai-ling.v2i1.16.

[10] S. Kopp, M. Brandt, H. Buschmeier, K. Cyra, F. Freigang, N. Krämer, F. Kummert, C. Opfermann, K. Pitsch, L. Schillingmann, C. Straßmann, E. Wall, R. Yaghoubzadeh, Conversational assistants for elderly users – the importance of socially cooperative dialogue, in: E. André, T. Bickmore, S. Vrochidis, L. Wanner (Eds.), CEUR Workshop Proceedings, volume 2338, 2018, pp. 10–17. URL: http://ceur-ws.org/Vol-2338/paper1.pdf.

[11] M. Trajkova, A. Martin Hammond, "alexa is a toy": Exploring older adults' reasons for using, limiting, and abandoning echo, 2020, pp. 1–13. doi:10.1145/3313831.3376760.

[12] K. Cyra, K. Pitsch, Dealing with 'long turns' produced by users of an assistive system: How missing uptake and recipiency lead to turn increments, in: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2017, pp. 329–334. doi:10.1109/ROMAN.2017.8172322.

[13] I. Lopatovska, H. Oropeza, User interactions with "alexa" in public academic space, Proceedings of the Association for Information Science and Technology 55 (2018) 309–318. doi:10.1002/pra2.2018.14505501034.

[14] V. Sklyarov, Hierarchical finite-state machines and their use for digital control, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 7 (1999) 222–228. doi:10.1109/92.766749.

[15] S. Zamanirad, B. Benatallah, C. Rodriguez, M. Yaghoubzadehfard, S. Bouguelia, H. Brabra, State machine based human-bot conversation model and services, in: Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, 2020, pp. 199–214. URL: https://hal.science/hal-03122974. doi:10.1007/978-3-030-49435-3_13.

[16] S. U. Meshram, Evolution of modern web services – rest api with its architecture and design, International Journal of Research in Engineering, Science and Management 4 (2021) 83–86. URL: https://journal.ijresm.com/index.php/ijresm/article/view/970.

[17] Json schema, Accessed: 2024-10-27. URL: https://json-schema.org/.

[18] S. Acosta, Anforderungen an das Sounddesign von User Interface Sounds zur audiogestützten Mensch-Maschine-Interaktion des Smart Companion 2 bei Menschen im dritten Lebensalter, Masterarbeit, Fachhochschule St. Pölten, 2023. URL: https://phaidra.fhstp.ac.at/detail/o:5631, abgerufen am 13. Januar 2025 unter: https://phaidra.fhstp.ac.at/detail/o:5631.

[19] T. Eder, Usability Evaluation of LLM-supported Alexa Skills for the Voice-controlled Scheduling of Robot Vacuum Cleaners and for Personal Emergency Response Systems for Fall Prevention and Support in Everyday Life for Elderly People: A Pilot Study, Masterarbeit, Fachhochschule St. Pölten, 2023. URL: https://phaidra.fhstp.ac.at/detail/o:5666, abgerufen am 13. Januar 2025 unter: https://phaidra.fhstp.ac.at/detail/o:5666.