VoiceCRM:
A voice-powered CRM

Cernera Federico, 1584227
Fernandes Eugenio, 1560403
Liberati Pietro Andrea, 1814440
Modenese Davide, 1665812
Quaranta Davide, 1715742

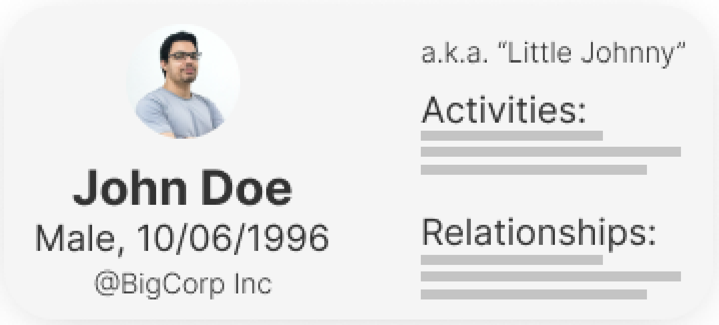
A.A. 2020/2021

Table of contents

Introduction

A **Contact-Relationship Manager**, or CRM, is a system in which a business or other organization administers its **interactions** with customers, typically by keeping track of a variety of information about them.

More generally, a CRM is a system to maintain **information about people** and to keep track of relationships, activities and notes about people. An example of such system oriented at a personal usage is Monica[1].



As shown later in the need finding section, CRMs are typically hard and slow to use, mostly because of the fact that certain operations require more steps than necessary. As an example, to register a new activity with a contact in Monica - as of the time of writing - the user needs to:

1. Open the CRM interface.
2. Possibly log-in.
3. Open the contact.
4. Reach the "Activities" section and click to create a new activity.
5. Fill the required fields and save.

This interaction, even though it involves everything necessary to accomplish the task, can feel overwhelming for simple and quick usages, thus compromising the overall usefulness of the CRM itself.

This project aims at **simplifying** some CRM tasks, by enabling users to perform them via their voice. For this purpose, a **prototype of a voice interface** was designed, developed and tested, following the Agile UCD (user-centered design) methodology.

The developed voice interface implements a personal CRM, and can be adapted with minimal effort to work with any other CRM that exposes an API.

This report describes the project in its multiple parts, and documents some of the activities performed during its realization.

[1] https://www.monicahq.com

Need finding

To identify the main needs of the CRM users a **questionnaire**[2] was created and delivered online to multiple places where potential users could be, including:

- Reddit communities about CRMs.
- The Reddit community Self-Hosted[3], where Monica CRM is popular.
- Various Telegram and Facebook groups about CRMs.
- Multiple Facebook groups about CRMs.

The questionnaire was structured to gather different types of information, based on the respondent's knowledge or experience with CRMs. Specifically, the questionnaire gathered information about:

- Respondent's knowledge/experience with CRMs.
- Problems or limits of the CRMs that the respondents have used.
- What can be improved in CRMs.
- Respondent's experience with voice assistants.
- Respondent's experience with voice-enabled CRMs, if any.
- Typical CRM tasks done with the voice.
- Device/medium preference for tasks, between PC, smartphone, tablet, voice.

As partly anticipated in the previous pages, the main needs extracted from this phase can be synthesized as follows:

- Reduce the number of steps to perform short-lived tasks.
- Simplify the execution of simple operations.
- Focus on the activities done with people.
- Simplify the definition of relationships between people.

Not every task is suitable for a voice interaction, namely those that are critical or where errors are not tolerated.

Results

Circa 170 people responded to the questionnaire, and the most interesting results can be summarized as:

- ~60% of the respondents have experience with CRMs.
- ~80% of which have used them for personal reasons.
- The most popular reasons of use are:
    1. To keep track of activities with people (~80%).
    2. To help remember information related to people (~70%).
    3. To keep track of relationships (~50%).
    4. To have reminders (~40%).
- The most used device for CRMs is the computer (~96%).
- The main problems of CRMs are:
    - Too many steps (~60%).

[2] https://forms.gle/8HG8JzGnSTZ2ribP6
[3] https://www.reddit.com/r/selfhosted/

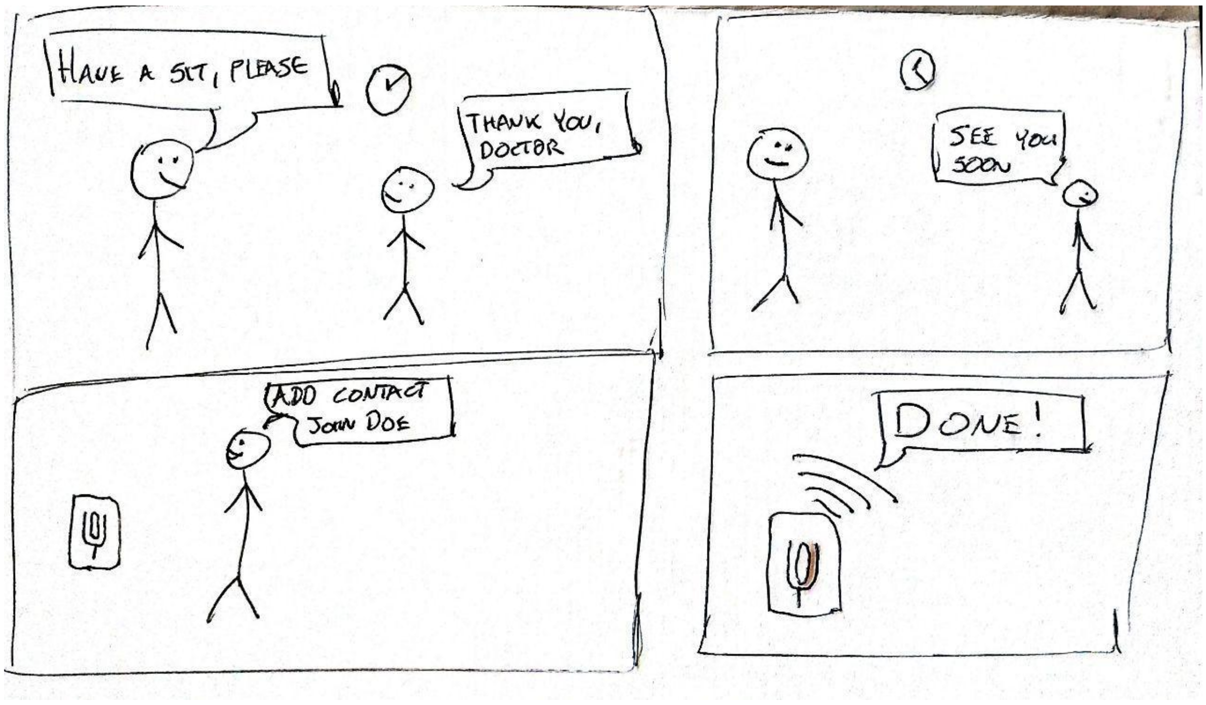
    - Lack of information flexibility (~40%).
    - Information overflow (~30%).
    - Bad UI (~30%).
- Some interesting comments on how to improve CRM interactions are:
    - Simplify the execution of common tasks.
    - Use of natural language to fill data.
- Only 4 people have used a voice-assisted CRM.

Tasks

After the analysis of the needs, a set of main tasks was extracted, and is described via the following storyboards.

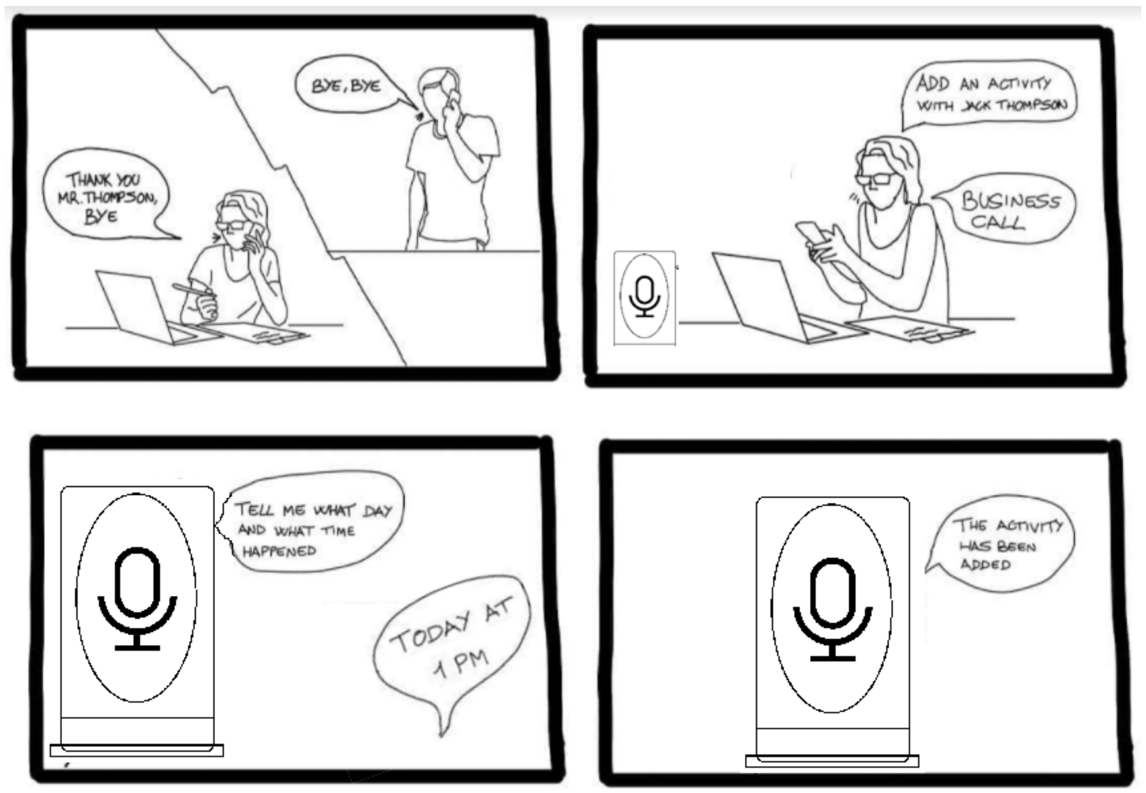
Task 1: new contact

The user can create a new contact by providing what they know about this person.



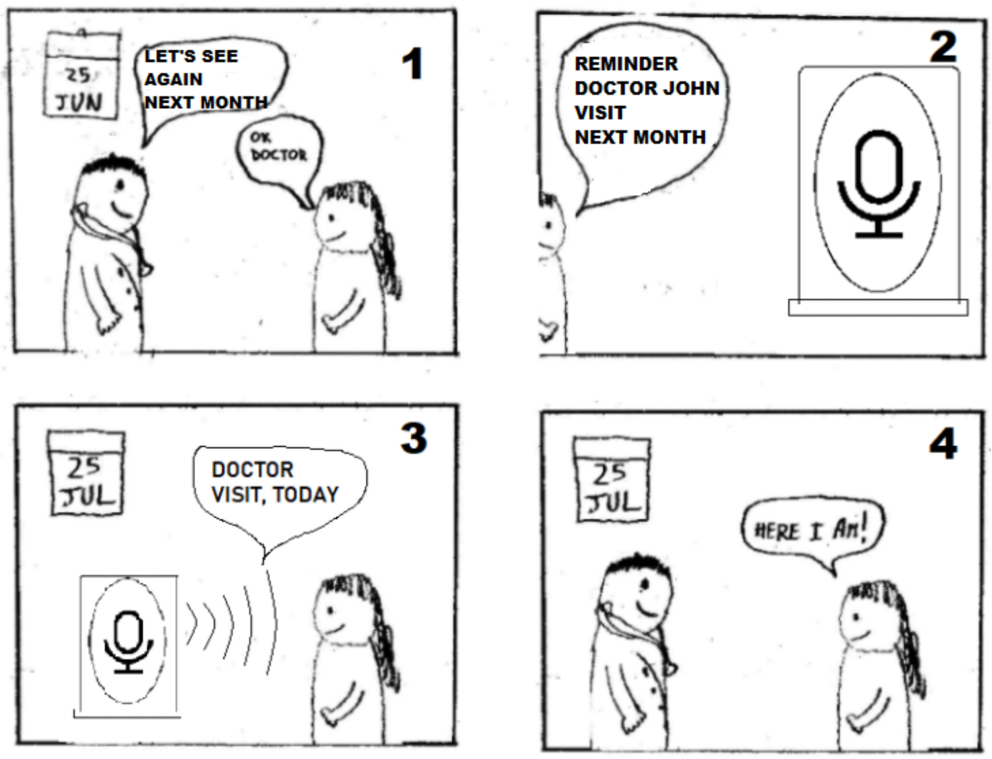
Task 2: new activity

The user can add a new activity done with a contact.


BYE, BYE
THANK YOU MR. THOMPSON, BYE
ADD AN ACTIVITY WITH JACK THOMPSON
BUSINESS CALL
TELL ME WHAT DAY AND WHAT TIME HAPPENED
TODAY AT 1 PM
THE ACTIVITY HAS BEEN ADDED

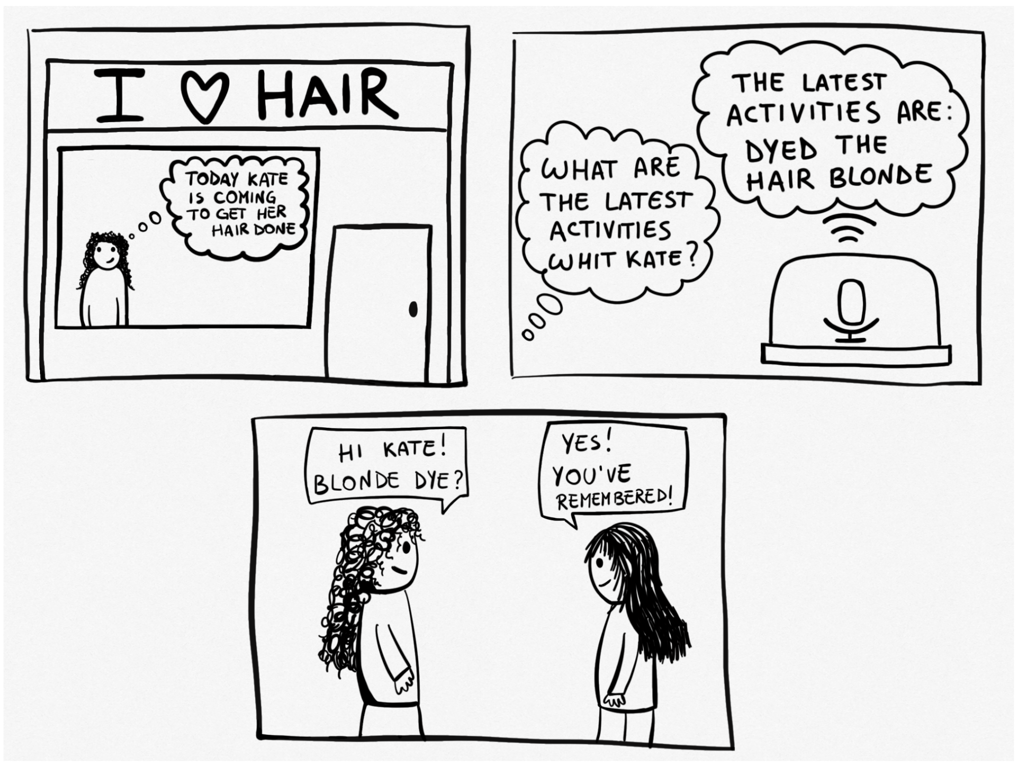
Task 3: new reminder

The user can set a new reminder related to a contact.


25 JUN
LET'S SEE AGAIN NEXT MONTH
1
OK DOCTOR
2
REMINDER DOCTOR JOHN VISIT NEXT MONTH
25 JUL
3
DOCTOR VISIT, TODAY
25 JUL
4
HERE I AM!

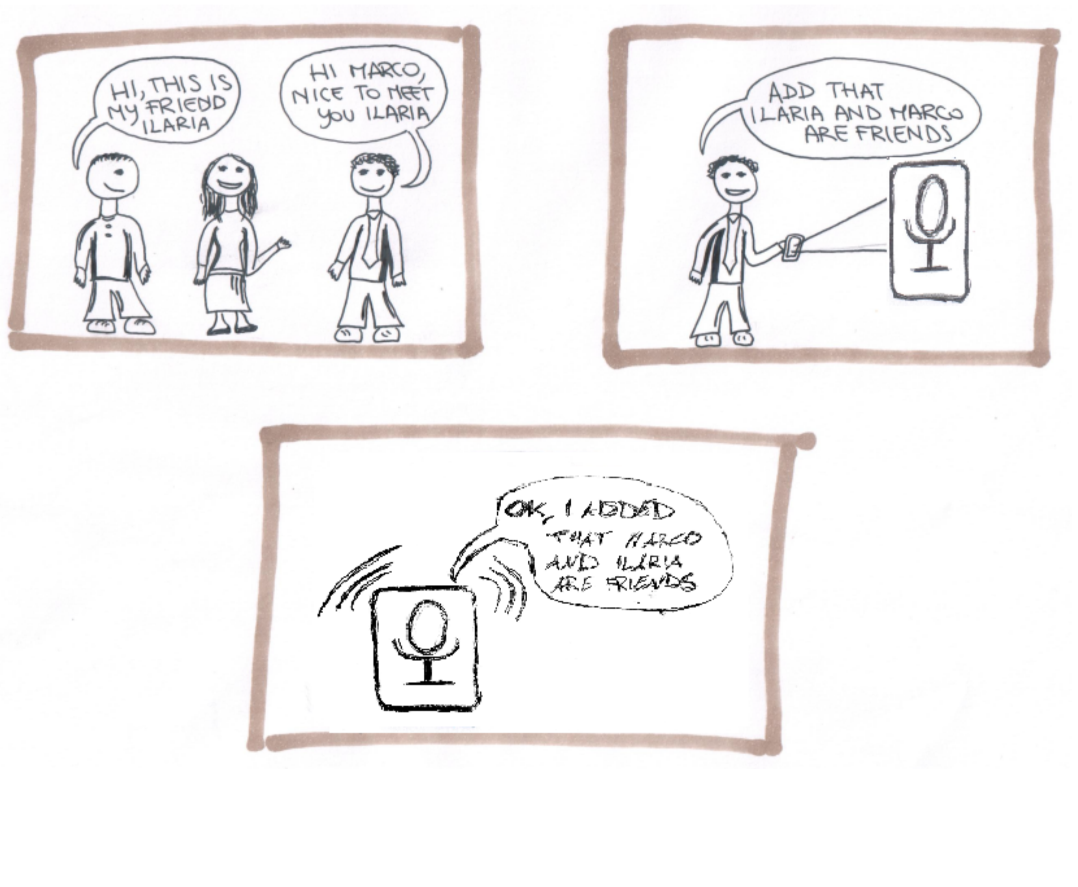
Task 4: list activities

The user can ask for a list of activities done with a contact.


I ♡ HAIR
TODAY KATE IS COMING TO GET HER HAIR DONE
WHAT ARE THE LATEST ACTIVITIES WHIT KATE?
THE LATEST ACTIVITIES ARE: DYED THE HAIR BLONDE
HI KATE! BLONDE DYE?
YES! YOU'VE REMEMBERED!

Task 5: new relationship

The user can add a relationship (friend, work, family) between two contacts.


HI, THIS IS MY FRIEND ILARIA
HI MARCO, NICE TO MEET YOU ILARIA
ADD THAT ILARIA AND MARCO ARE FRIENDS
OK, I ADDED THAT MARCO AND ILARIA ARE FRIENDS

Prototyping and development

Pre-totyping

Given the nature of the interface, it is not possible to rely on prototyping tools and artifacts such as animated paper prototypes, in order to **understand how the user naturally approaches the vocal interface**.

For this reason a “pre-totyping” approach was taken into account. Specifically, we acted as a voice assistant, to simulate and evaluate users’ interactions.

Context

Given a context, tests for all the tasks reported in the previous pages were conducted. To personalize the test, the user was given freedom to decide the names and other details of the test (e.g. activities, what to remind, contact details).

Task 1a: “You want to add a person in the system. You know name, surname, age”

Task 1b: “You want to add a person in the system. Only name and surname known.”

Task 2: “You want to register into the system that you just had a phone call with a contact of yours”

Task 3: “You want to make the system remind you of something on the 20th of June”

Task 4: “You want to have the list of the latest activities done with a contact of yours”

A test transcription of the pre-totyping phase can be found in the appendix.

Results

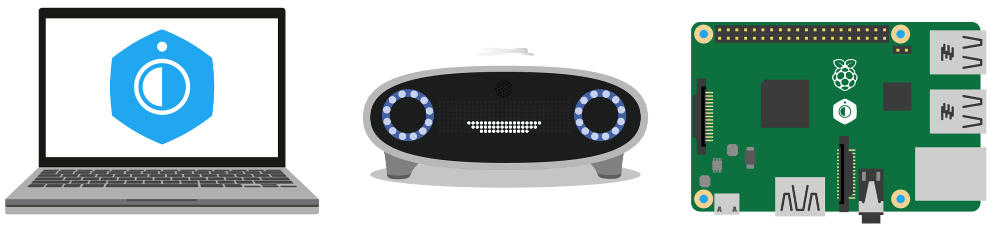
The main insights gathered from this phase can be summarized with these points:

- The user naturally tends to mention the contact whose task is about.
- In case the user does not mention them, further questions are needed, thus producing a longer and guided interaction.
- The interaction needs some form of flow control, mostly for error recovery.
- If the user speaks in a sufficiently structured way, some missing information can be deduced from the interaction.

Prototyping

After gathering the initial information from the pre-totyping phase - namely an idea of possible utterances and intents - it was possible to start developing a real prototype that the users could use.

The chosen technology was **Mycroft**[4]: an open source and **privacy focused voice assistant** that can be deployed on various platforms, such as personal computers, boards like the Raspberry Pi, or apposite hardware (like Mycroft Mark I).



Mycroft concepts

Mycroft can be seen as a modular suite of software; the most important components are the STT (speech-to-text) engine, the TTS engine (text-to-speech) and the intent parser.

The default STT engine is **Google STT**, but Mycroft supports Mozilla's DeepSpeech, IBM Watson and wit.ai. The default **TTS** engine is Mycroft's Mimic (local), but other easily configurable engines are Microft's Mimic2 (cloud based) and Google TTS.

The **intent parser** takes the output of the STT engines and extracts the *intent* of the user's speech (**utterance**), based on an intent definition created by the developer, which is a collection of phrases that are in some way representative of the intent. Intents may contain **entities**, namely placeholders for the data that the user provides, such as contact information, dates, names, etc. After an intent is triggered, the corresponding handler function is called.

The **middleware** between the skills and the engines is composed by the Mycroft Core components and APIs. Mycroft and its skills are written in **Python 3**.

[4] https://mycroft.ai

Methodology

The development methodology that was used in this project is **Agile UCD** (User-Centered Design), characterized by a continuous feedback loop between design and development, with focus on user needs, usability tests, shared team knowledge and frequent releases.

Tests were performed by making multiple users interact with the interface, after giving them a context, namely a set of information including a description of what a CRM is, what this interface allows to perform, and a progressive list of tasks.

The general setup for the tests was:

- **Introduction**: the user is introduced to the concept of usability test and how it is conducted.
- **Context definition**: the user is given a task, with the preliminary information needed to accomplish it.
- **Execution**: the user executes the task and relevant observations are noted.

This chapter documents the performed iterations, as well as the changes that they introduced.

Iteration 1

In this first iteration an *Earliest Testable Product*[5] was developed, that featured - for each task:

- A minimal set of intents.
- A minimal set of phrases to be pronounced by the interface.
- Basic error handling.
- Basic state management.

During this phase, data models and development strategies became more defined, as described in the following subsections.

Data overview

A **local contacts database** was created to as underlying data layer; specifically:

- Each **contact** is identified by a name and surname, and is uniquely identified by a nickname, if any.
- Contacts can have **optional information**: in this prototype birth date and gender are supported.
- Contacts have a list of **activities** related to them, composed by:
  - The description of the activity.
  - A date and time.
- Contacts can have **relationships** between them, defined by the type (e.g. friendship, spouse, ...).
- Contacts also have a list of **reminders** related to them, composed by:
  - What to remind.
  - A date and time for the reminder.

After the user sets a reminder, the preinstalled external skill Reminders[6] is called, to which is delegated the behavior of “waking up” at the time set by the user, and to say the reminder out loud.

Common development framework

To easily manage the structure of the needed tasks, a framework to base each task on has been developed. The framework imposes a **task state management standard**.

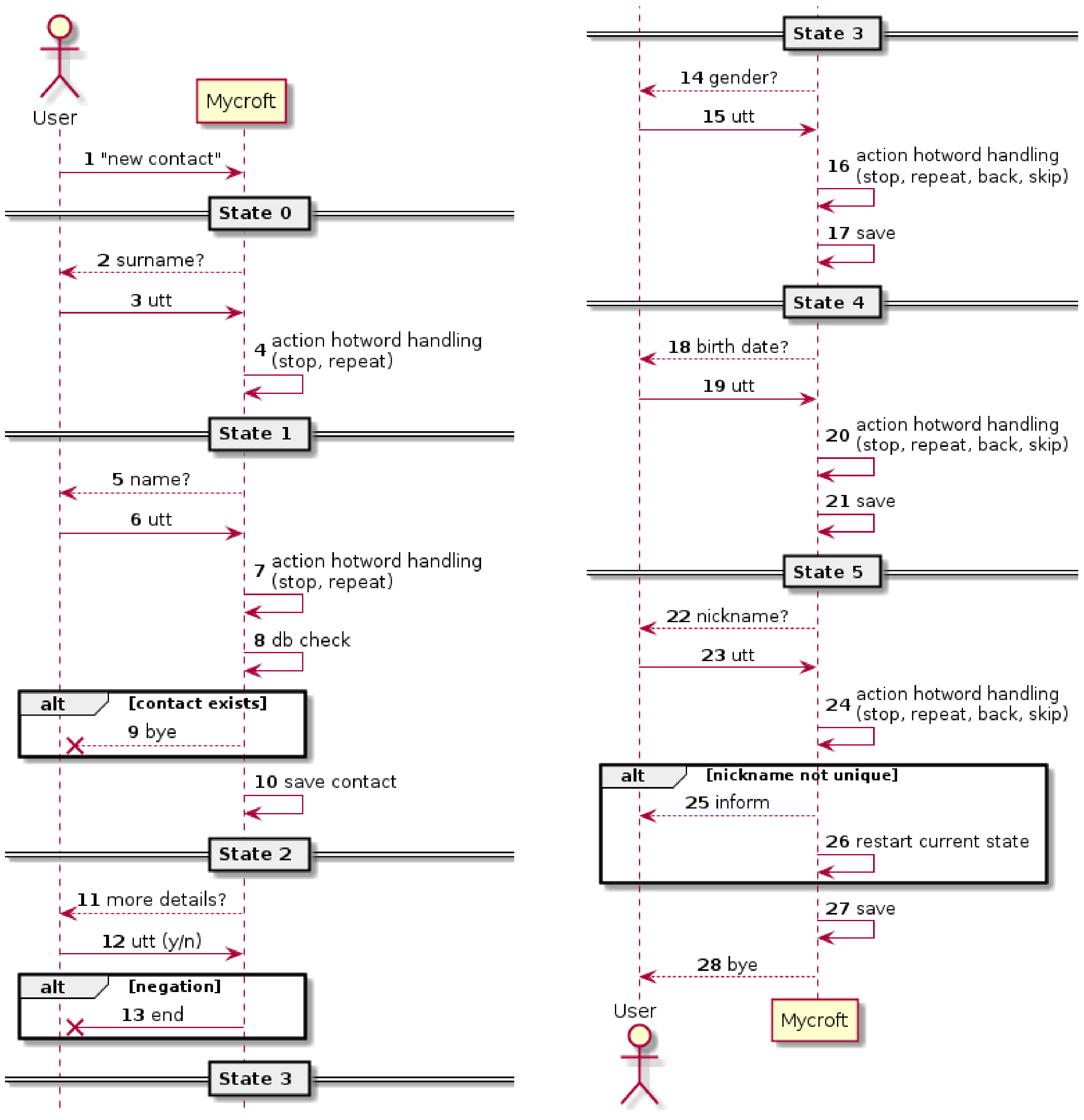
A state represents an interaction step in which the user provides information, requested either by the user itself or by the interface. The task is then divided into a sequence of states, that are handled in an uniform way.

[5] https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp
[6] https://github.com/MycroftAI/skill-reminder/

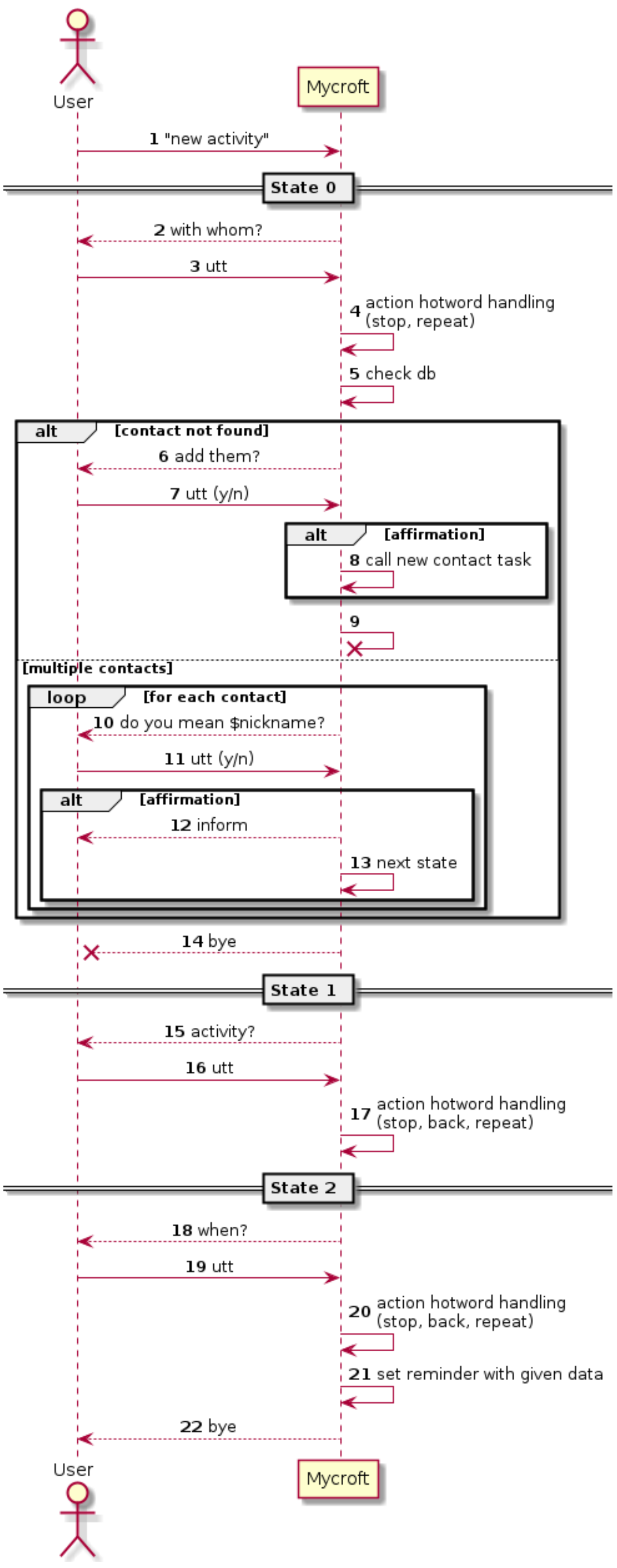
Diagrams

A description of the structure of the interaction is synthesized by the following diagrams:

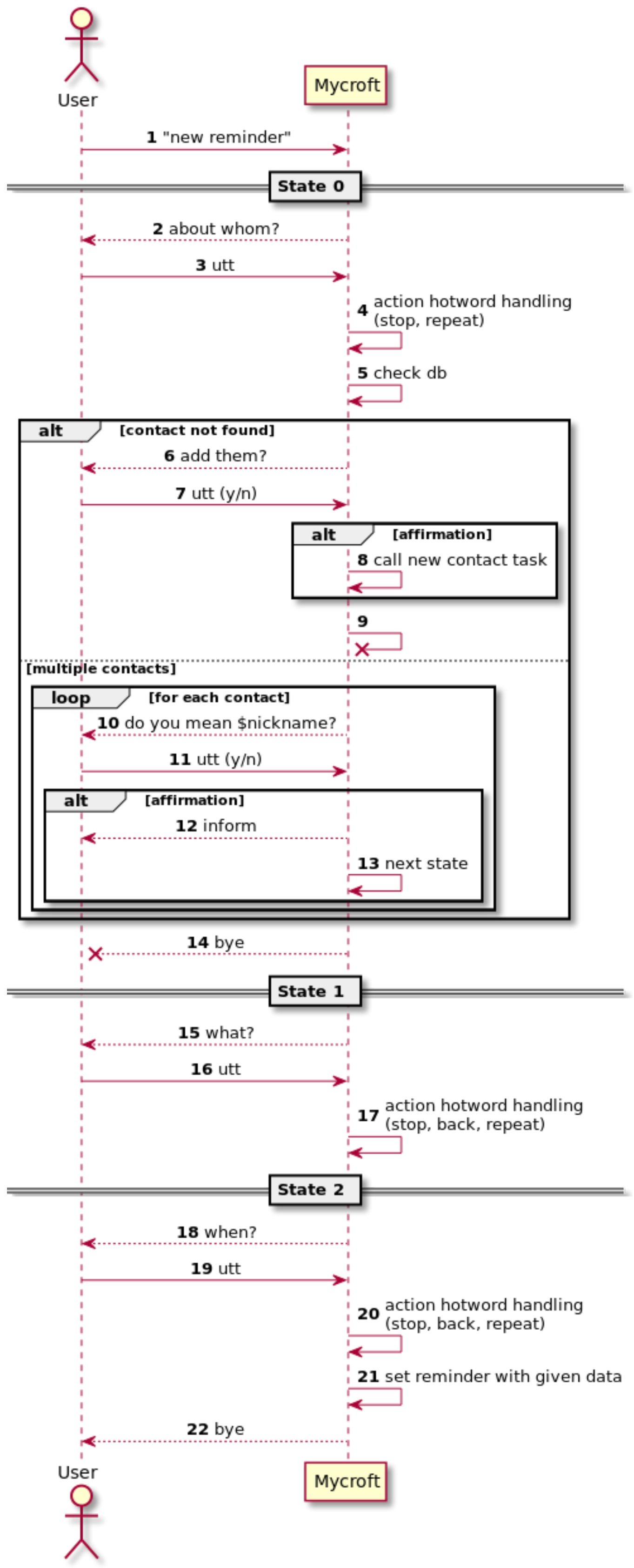
New contact



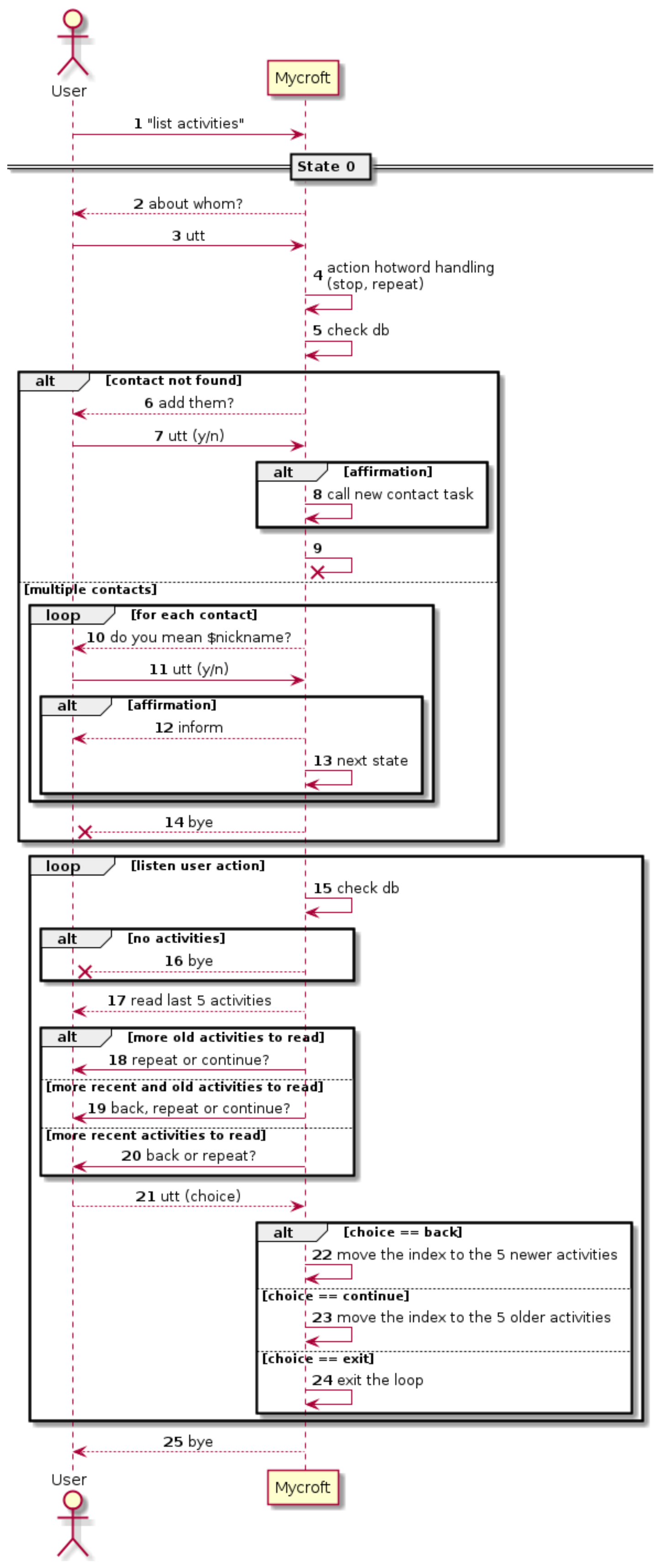
New activity



New reminder



List activities



In the context of the diagrams, the phrase “action hotword handling” refers to the action of matching the utterance against a list of **special words** that have a defined meaning in that moment, such as “stop”, “repeat”, “back”, “skip”. These actions enable the user to control the interaction flow.

More precisely, the behavior of each action word is:

- **Stop**: discard the current status and stop the interaction.
- **Repeat**: repeat the current question or generally the current state.
- **Back**: go to the previous question.
- **Skip**: discard the current question and go to the next one.

There are multiple expressions that are mapped to the same behavior: for example the user can express to skip by saying “I don’t know”, “skip”, “I don’t care”, etc.

Some actions are not permitted for certain steps: for example when creating a contact the user cannot skip questions about the minimal data needed, like name and surname.

To focus on representing the core of the tasks, all the logic of handling the action words has been synthesized by the sole phrase “action hotword handling”.

Test results analysis

The first test iteration showed some problems and made other styles of utterance emerge; in particular:

- Some utterances were not recognized.
- In some cases the interface didn’t give enough time to answer.
- In some situations the user naturally tried to express multiple data in a single utterance, but the interface didn’t handle it.
- The overall interaction seemed to be too robotic.
- Bugs emerged in different parts of the code.

After having analyzed and compared the test results, the above range of problems were addressed in the next iteration.

Iteration 2

In this iteration, a broad range of **improvements** were done, specifically:

- The intent file definition for each task was enhanced to allow multiple variations of the same phrases, and other phrase constructions learned from the test results.
- Better variations of pronounced phrases were added, to make the interaction feel more friendly, varied and context-aware.
- *Compact intents* were supported, to extract multiple data in a single utterance.
- The code has been refactored and improved.

Moreover, the New Relationship task was implemented.

In the scope of this document, ***compact intents*** are defined as utterances that also contain user data (entities) that need to be extracted and processed. As an example, below is reported an intent file definition for the New Activity task:

```
(|add|create|register|another|record) (|a) new activity
(add|create|register|another|record) (|a|an) (|new) activity
(add|create|register|another|record) (|a|an) (|new) activity {DateTime} with
  ↲ {Person}
```

The first two lines define normal intents without entities; the last one contains the `DateTime` and `Person` entities. The `|` (pipe) character represents optionality.

The **most relevant improvements** to the tasks flow are:

New contact task

- A bug that did not ask for the nickname was solved.
- The overall interaction was made to feel more friendly.
- Compact intents were added.

New activity task

- After the creation of a non existing contact, the task now continues.
- Compact intents were added.

New reminder task

- Compact intents were added.

List activities task

- The activities are now read from the most recent one to the least recent, maintaining the groups of 5, as shown in the diagram.
- Dates are now read in a more human-friendly way.
- Compact intents were added.

Generally, improvements that are not strictly related to a task are:

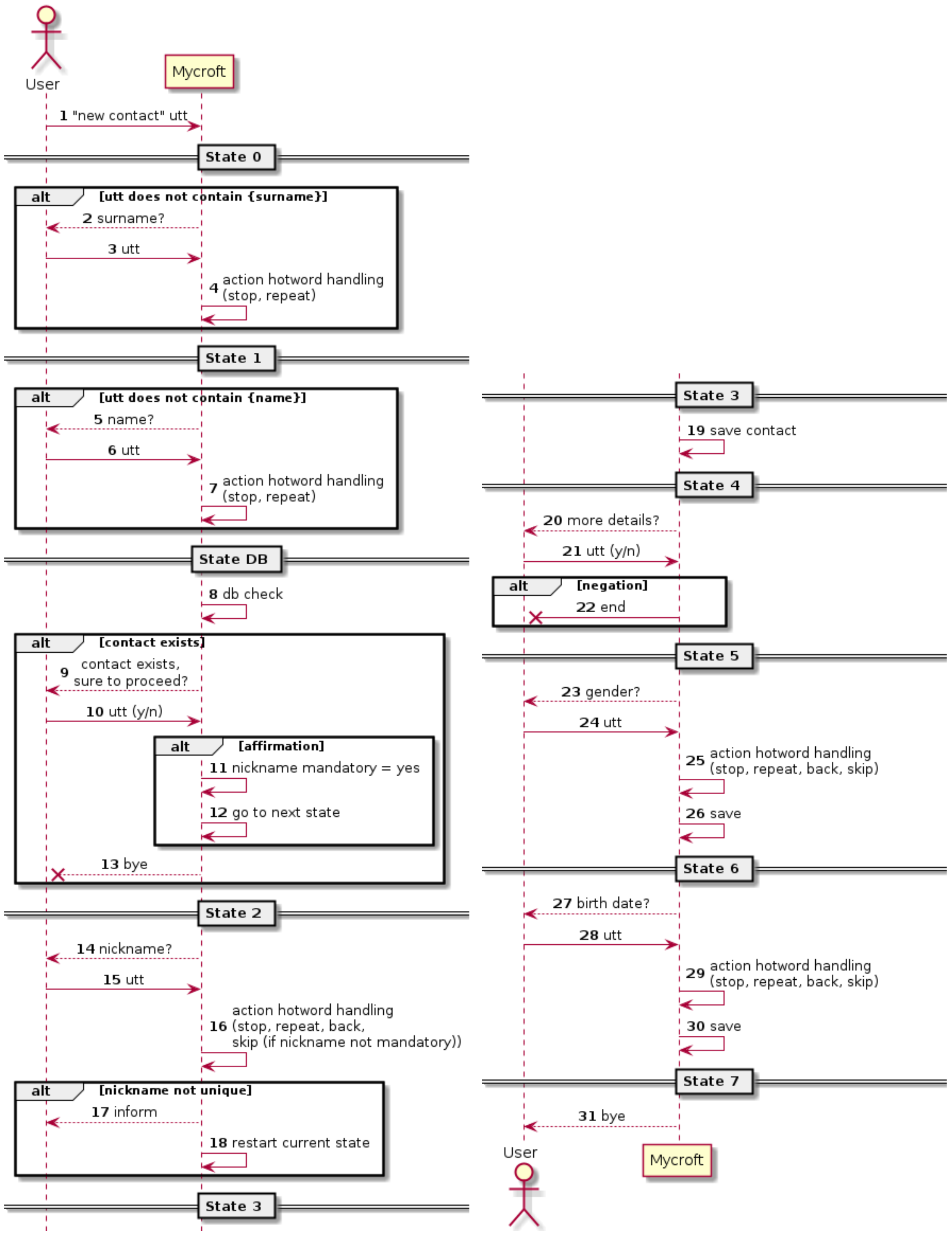
- Date parsing was improved and better checked against errors.
- Better handling of empty utterances.
- Better handling of exceptional behavior like “stop” requests.

At the end of this stage, the interaction felt more natural, and in certain cases less steps were needed to complete a task.

Diagrams

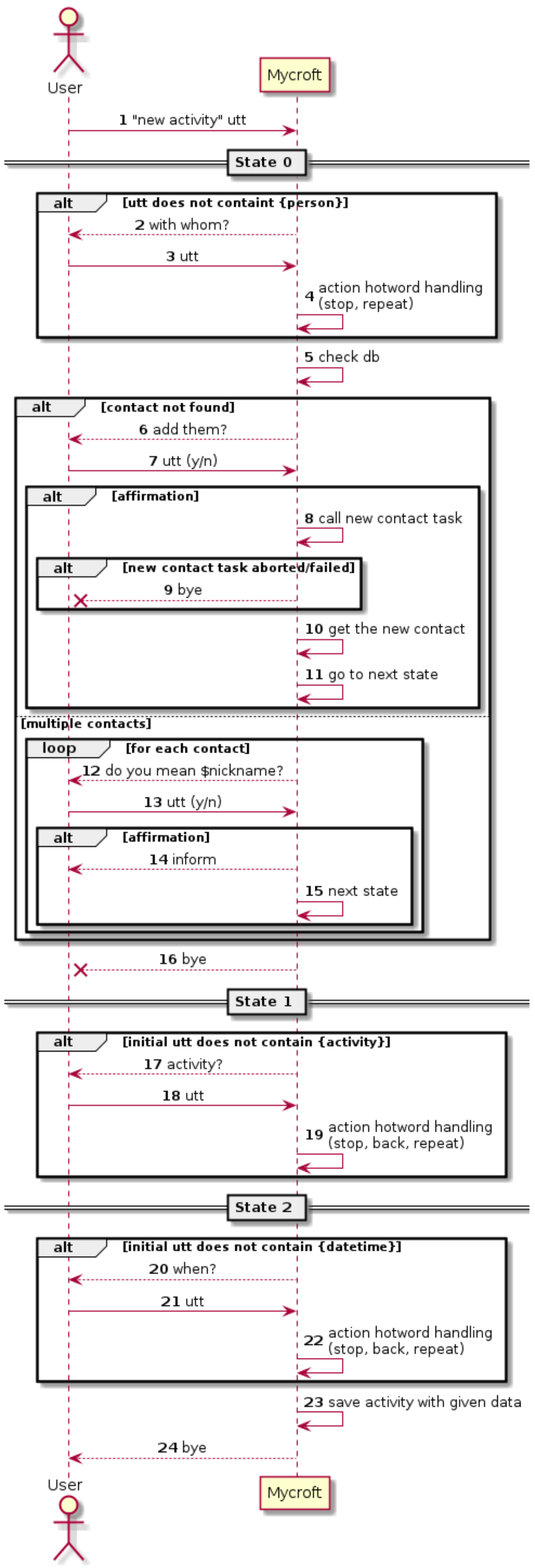
The following diagrams describe the new task flows.

New contact

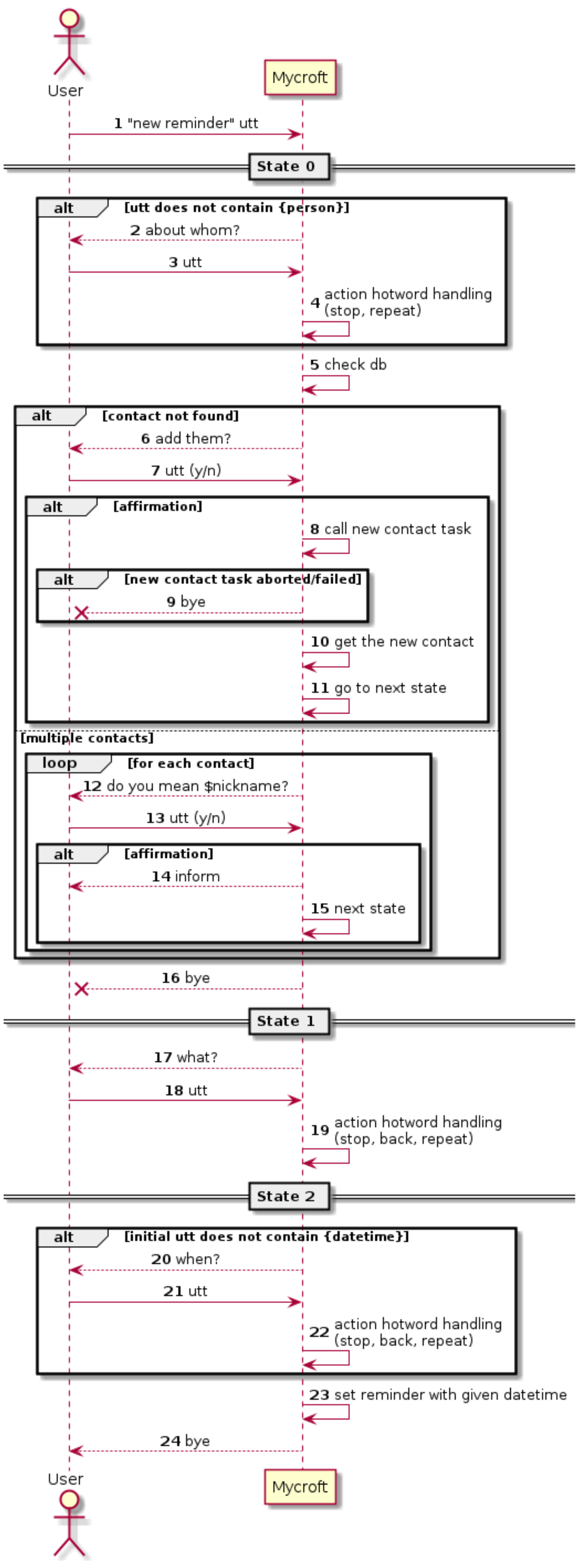


User
Mycroft
1 "new contact" utt
State 0
alt [utt does not contain {surname}]
2 surname?
3 utt
4 action hotword handling (stop, repeat)
State 1
alt [utt does not contain {name}]
5 name?
6 utt
7 action hotword handling (stop, repeat)
State DB
8 db check
alt [contact exists]
9 contact exists, sure to proceed?
10 utt (y/n)
alt [affirmation]
11 nickname mandatory = yes
12 go to next state
13 bye
State 2
14 nickname?
15 utt
16 action hotword handling (stop, repeat, back, skip (if nickname not mandatory))
alt [nickname not unique]
17 inform
18 restart current state
State 3
State 3
19 save contact
State 4
20 more details?
21 utt (y/n)
alt [negation]
22 end
State 5
23 gender?
24 utt
25 action hotword handling (stop, repeat, back, skip)
26 save
State 6
27 birth date?
28 utt
29 action hotword handling (stop, repeat, back, skip)
30 save
State 7
31 bye
User
Mycroft

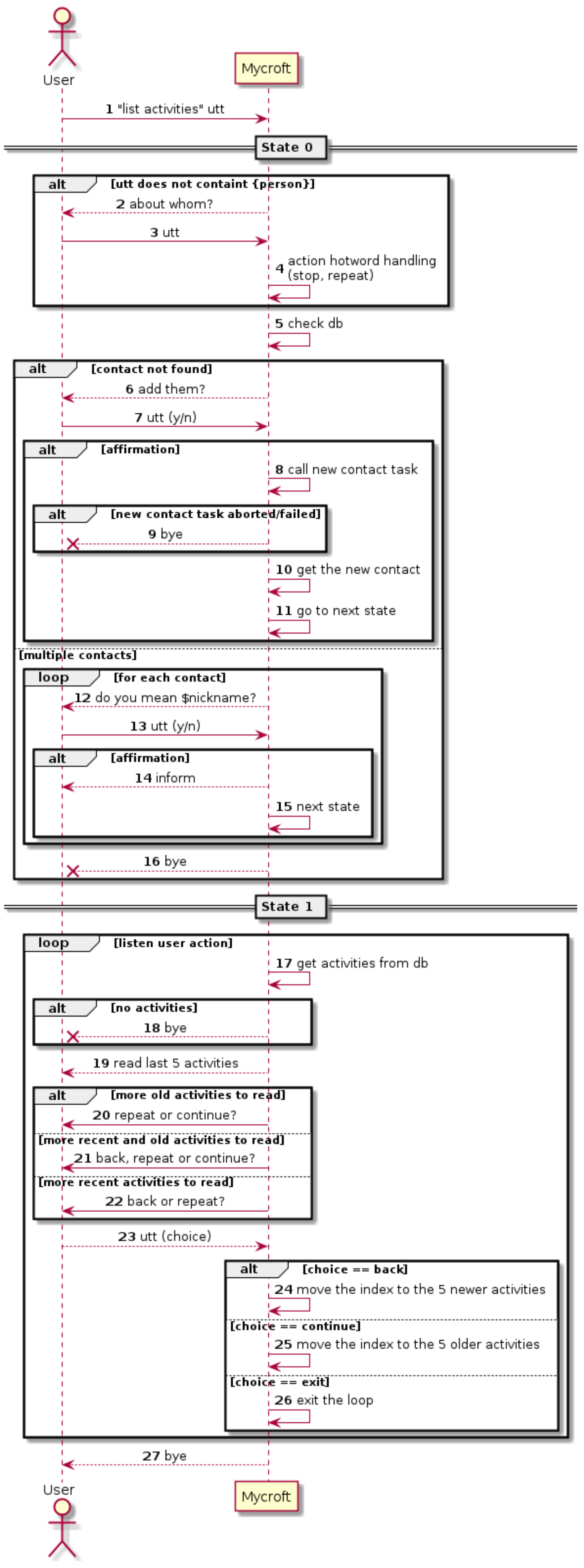
New activity

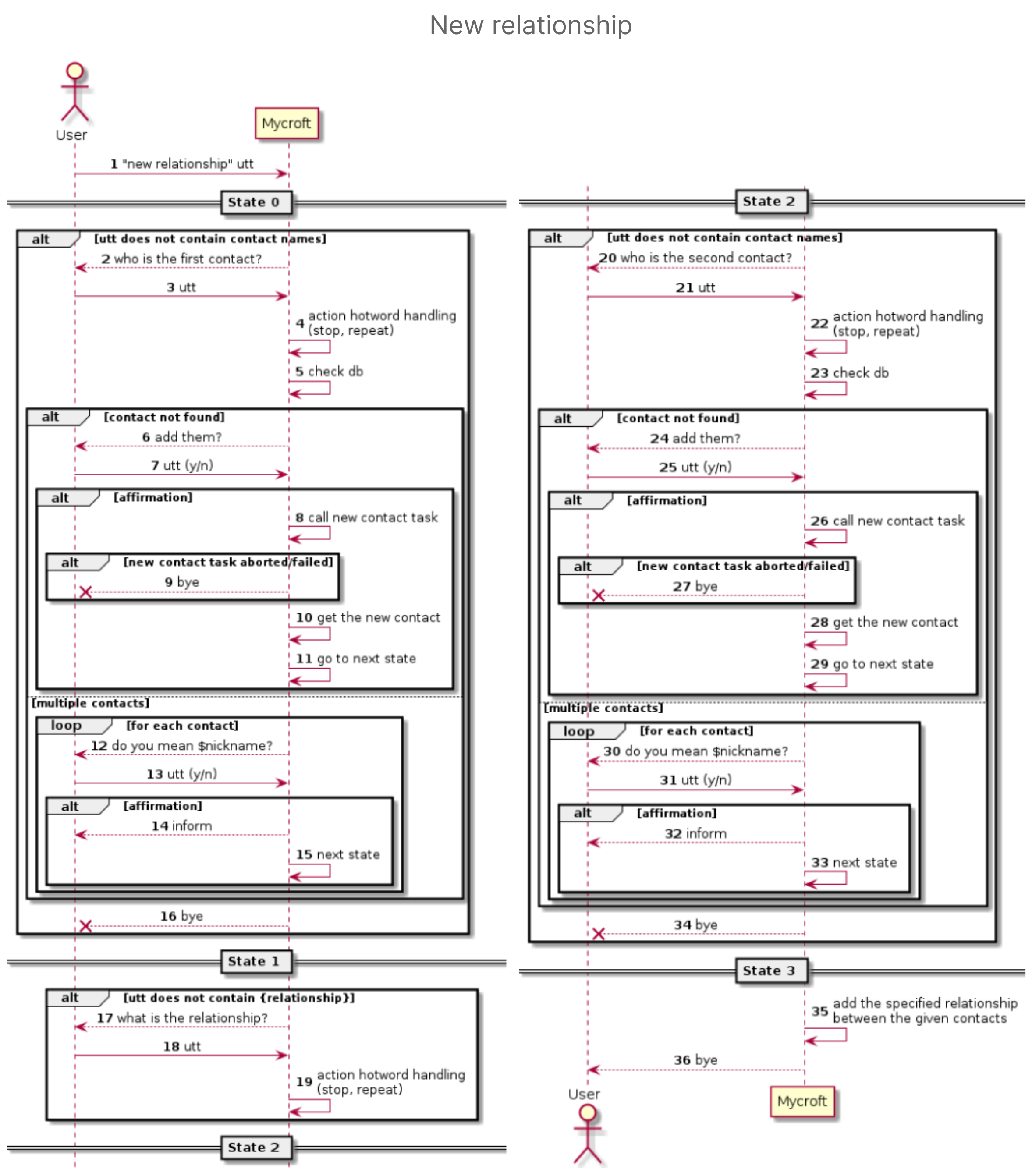


New reminder



List activities





Test result analysis

Tests performed in this iteration showed that:

- Some utterances were not recognized.
- Data extracted from utterances was sometimes wrong.
- The flow control was not working in some cases.
- Contact search was not working in some cases.

Iteration 3

The third iteration featured **improvements** to the:

- Intent parsing and data extraction.
- Flow control.
- Handling of stopwords and other undesired content in utterances.
- Data validation.
- Error prevention and handling.
- Phrases pronounced by the interface.

Specifically, the most significant improvements can be summarized as follows:

Intent parsing

The intent parsing has been improved by migrating from Padatious[7] to a new intent parser called **Adapt**[8], that allows more flexibility and works better in case of less structured utterances, where the user may or may not include entities.

Adapt is keyword-based and intent definitions are made by **regular expressions** that may contain extractable entities. By fine tuning keywords and expressions it was possible to land the user to the correct task, even if the utterance was not correctly structured.

If the utterance - instead - is well structured per regex definition, data will be directly extracted and a shorter task execution will be possible.

Below is reported an example Adapt intent file for the New Activity task:

```
.* (?P<DateTime>(monday|tuesday|wednesday|thursday|friday|saturday|sunday|yesterday|tomorrow))
.* with (?P<Person>.*)
.* on (?P<DateTime>.*) with (?P<Person>.*)
.* with (?P<Person>.*) on (?P<DateTime>.*)
.* with (?P<Person>.*)
```

As it is possible to see, with structured utterances it is possible to extract the `DateTime` and `Person` entities; in case of non-structured utterances, if the user pronounced the **"activity" keyword**, the correct task will be started as well.

Undesired words filtering

The stopword/undesired words filter has been vastly improved, so that common user errors are transparently handled.

As an example, if the utterance for telling the new contact's name contains variations of "the name is John", only "John" passes through the filter.

[7] https://github.com/MycroftAI/padatious
[8] https://github.com/MycroftAI/adapt

Data validation

More validation edge case were covered and handled, particularly:

- Denial of self-relationships.
- Denial of multiple family relationships between two contacts.
- When creating a new activity, weekdays are always treated as past, to overcome Mycroft’s default for the future.

Test result analysis

Tests conducted on this iteration **validated** the modifications stated above, and only suggested minimal wording improvements to the pronounced dialogs and further minimal tuning of the intent recognition strategies.

As an example, it has been fixed a speech recognition issue for which sometimes the word “contact” was incorrectly recognized as “contactor”; similarly, when asking for the gender of a person, the words “mail”, “main”, “Maine” have been mapped to “male”.

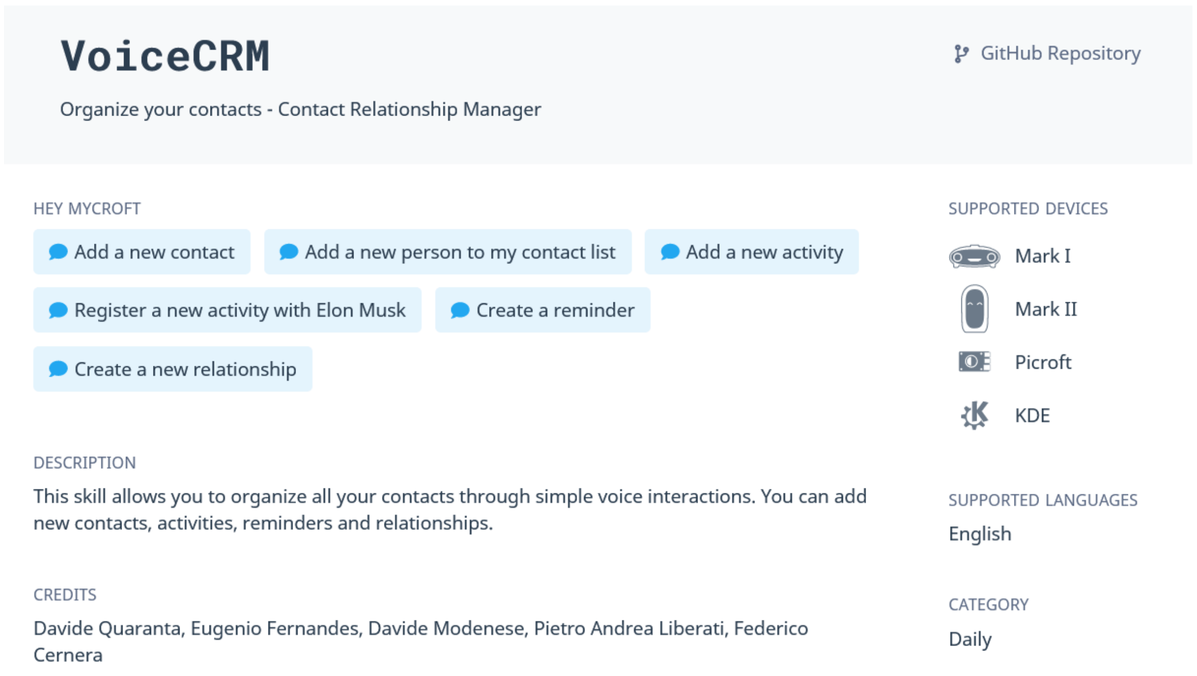
With this version, the interaction is satisfactory, less error-prone, sufficiently friendly, and the user is more aware of the actions performed by the interface.

Conclusions

While developing a vocal interface, multiple issues can arise, such as the correct extraction of user entities, as well as low-level or external issues like microphone quality, user accent, tone of voice and environment noise.

The **main challenge** of this project - especially during the third iteration - has been to mitigate the effects of these issues, for example by mapping common STT or pronunciation errors to expected behaviors and by **fine-tuning the keywords** to increase the chances of starting the right task even in case of disordered and **unstructured utterances**.

This screenshot represent how a user that owns a Mycroft device would see the specification of the VoiceCRM skill, on the Mycroft Marketplace[9]. Like most skills, the description includes some **examples of valid utterances**, so that the user can quickly gather a **hint** at what the skill supports and how to interact with it.



VoiceCRM

Organize your contacts - Contact Relationship Manager

GitHub Repository

HEY MYCROFT

- Add a new contact
- Add a new person to my contact list
- Add a new activity
- Register a new activity with Elon Musk
- Create a reminder
- Create a new relationship

DESCRIPTION

This skill allows you to organize all your contacts through simple voice interactions. You can add new contacts, activities, reminders and relationships.

CREDITS

Davide Quaranta, Eugenio Fernandes, Davide Modenese, Pietro Andrea Liberati, Federico Cernera

SUPPORTED DEVICES

Mark I

Mark II

Picroft

KDE

SUPPORTED LANGUAGES

English

CATEGORY

Daily

[9] https://market.mycroft.ai/skills

Appendix

Pre-totyping transcription

A transcription of a pre-totyping test is reported, in which the interface is referred to with the name *Pino*. Lines in *italic* are pronounced by the mocked interface.

Task 1: new contact

Well known person:

1. Pino, add John Smith
2. *Is the name John?*
3. Yes
4. *Is the surname Smith?*
5. Yes
6. *Good! Do you remember the age?*
7. 30 years old
8. *I've added John Smith, 30 years old!*

Not well-known person:

1. Pino, register a name
2. *Do you want to add a new contact?*
3. [...] to remind me [...]
4. *Do you want to set a reminder?*
5. Yes
6. *Who is this reminder about?*
7. Vincent Plumber.
8. *I don't know any Vincent Plumber. Do you want to add it?*
9. Yes
10. *Is the name Vincent?*
11. Yes
12. *Is the surname Plumber?*
13. Yes
14. *Good! Do you remember the age?*
15. No
16. *I've added Vincent Plumber. What was the reminder?*
17. Remind me to call him tomorrow.
18. *I've set a reminder related to Vincent Plumber: "remind me to call him tomorrow".*

This test is interesting because it contains a user error that leads to a wrong task, which requires the right task to continue; the execution of the nested task is, finally, successful.

Task 2: new activity

1. Pino, register a conversation with Julia Miller.
2. *Do you want to register a conversation done with Julia Miller?*

3. Yes
4. *How did you speak to Julia?*
5. On the phone
6. *What do you want to note about it?*
7. We talked about papers to send to the accountant.
8. *Good! I've registered your phone call with Julia Miller.*

Task 3: new reminder

1. Pino, add a reminder
2. *Which contact is it about?*
3. Dave Holland
4. *What is the reminder?*
5. Call
6. *When do I have to remind you?*
7. At 18:00
8. *Good! I will remind you to call Dave Holland at 18:00*

Task 4: list activities

1. Pino, tell me the latest activities with Marta Stewart
2. *21 May: sea; 10 May: lunch*