Soft Comput (2016) 20:3321–3334
DOI 10.1007/s00500-015-1709-2

METHODOLOGIES AND APPLICATION

Fuzzy system to adapt web voice interfaces dynamically in a vehicle sensor tracking application definition

Guillermo Cueva-Fernandez[1] · Jordán Pascual Espada[1] · Vicente García-Díaz[1] · Rubén González Crespo[2] · Nestor Garcia-Fernandez[1]

Published online: 15 May 2015

Abstract The Vitruvius platform is focused on vehicles and the possibility of working with their multiple sensors, and the real-time data they can provide. With Vitruvius, users can create software applications specialized for the automotive context (e.g., monitor certain vehicles, warn when a vehicle sensor exceeds a certain value, etc.), with the help of fuzzy rules to make decisions. To create applications, users are provided with a domain-specific language that greatly facilitates the process. However, drivers and some passengers cannot create applications on the fly since they need to type to accomplish such a goal. In this paper, we present an adaptive speech interface to allow users to create applications by only using their voice. In addition, the application is based on fuzzy rules to suit the level of experience of users. The application provides an interface that is balanced between the amount of work users have to do and the help the system provides based on the knowledge and ability of each potential user.

Keywords Fuzzy logic · Fuzzy decision making · Vehicle sensor · Tracking · Speech interface · Adaptive

Communicated by V. Loia.

✉ Guillermo Cueva-Fernandez
guillermo.cueva.fernandez@gmail.com

[1] Department of Computer Science, University of Oviedo, Asturias, Spain

[2] College of Engineering, Universidad Internacional de La Rioja, Madrid, Spain

1 Introduction

In the recent years, we have had a great increase of technology in different areas. The area of vehicles and transportation has experienced great technological advances. Among those advances are sensors and communication systems. Now the people can use a wide range of devices to extract information in real time about the vehicle. Most new vehicles are equipped with a lot of diriment sensors (location, speed, gasoline, rpm, CO_2, etc.) whose information can be very useful for a lot of software applications. Previously, our research work was focusing on the Vitruvius platform Cueva-Fernandez et al. (2014, 2015).

Vitruvius platform receives and manages data about vehicle sensors in real time and allows users to generate applications using a client. For generating applications, Vitruvius clients use a specific XML language. The language allows defining the most important functionalities of the application in an easy way, without programming knowledge. The applications that Vitruvius is able to generate are related to the vehicle data, maps and notifications managers. For example, the users can generate an application such as to follow in real time the position of a few vehicles on a map, to show the speed that a vehicle has gone on each part of a route, to get a notification every time that a vehicle surpasses a certain level of CO_2 emission, etc.

Vitruvius platform has opened a lot of possibilities for users and drivers. The applications defined by the users can be very useful in a lot of areas and situations, and the users are free to generate the applications that they need at any moment. However, the definition of applications using an XML language is not a very good alternative for a system which is mainly designed for drivers. One of the most beneficial aspects of Vitruvius is the possibility to generate applications on demand in real time to cover unexpected sit-

uations. But drivers cannot define an application using XML when they are driving; therefore, they need another system that is able to define applications while driving.

One possible solution to the problem is the use of voice interfaces. The drivers could use a voice interaction system to generate the applications in real time while driving. The use of a classic voice interaction system would not be very efficient in Vitruvius platform. The voice interaction systems cannot be easy to understand for all users in all situations, and it is likely that the interaction process will take several different steps some of which the user could feel confused by. The usability is a key factor in voice interaction systems; for a good usability, the system has to considerer a lot of aspects about the user experience and behavior.

In many cases, the voice interaction applications are designed based on a unique user profile, or on few user profiles, for example: beginners, standards, experts. In rapidly changing environments, the skills and the attention of the user are not always the same. For example, while driving there could be a lot of externals factors that can affect the attention of the user, the traffic, emotional state, the weather, actions of other occupants of the vehicle, etc. The static included in most voice interfaces is not able to adapt the interaction dynamically to the user. If the application could adapt the interaction dynamically, a better user experience could be achieved. For example, some users may be very efficient in some parts of the interaction process and they can use complex voice interaction systems. In other parts of the process though, it may be more difficult and those users would need more help provided by the system. As we are dealing with a very complex and unpredictable system, it is very difficult to foresee the way in which the user will understand the interaction system.

The purpose of this research work is to design an intelligent system able to improve the user experience and effectiveness in the use of voice interaction interfaces. This system is specially designed for a wide range of different user profiles and for complex highly variant conditions which can affect user's skills and concentration. The proposed system is based on an analysis of the user behavior while the user is using the voice interaction interface. Through the user behavior, the system will be able to detect if the user is able to use properly the voice interface or not; for this purpose, the system uses a set of fuzzy rules (Dutta and Chakraborty 2015). The result will be obtained in real time every time the user speaks. Based on the result of the analysis, the system will change dynamically the properties of the voice interaction system, for example, giving more or less detail in the instructions, or giving more or less response options.

Fuzzy logic is a way to define an approximate reasoning (Bouchon-Meunier and Valverde 1999), rather than one that is fixed and exact. In real life, there are a lot of situations in which it is not possible to clearly differentiate between two values (e.g., what it is big or small or what is good or bad), and it is not possible to use binary alternatives to make any decision. Fuzzy logic can be used in those cases in which it is necessary to deal with approximate (ranged between 0 and 1) rather than fixed and exact reasoning (Zadeh 1965; Kóczy 2006).

There are a lot of intelligent systems that apply a great level of successful fuzzy approaches, and many of these systems analyze big sets of data aiming to detect certain situations. In most of the cases, these situations could not be detected using traditional logic or algorithms because these situations depend on a lot of factors, with relative importance levels, and some of them cannot be absolutely bounded. For example, Ghafoor et al. (2015) worked on an intelligent approach to discover common symptoms among depressed patients using data mining techniques on a depression database containing 5,954 records. Authors make an inference that shows fuzzy concept is beneficial in these types of situations.

Lledó et al. (2015) classified physiological reactions of patients to automatically and dynamically adapt an assistant robot using a fuzzy classification method. Other examples include Silva and Serra (2014), where fuzzy logic is used to create an intelligent genetic fuzzy inference system for speech recognition based on discrete cosine transform. Since differentiating among the possible levels of knowledge of users who use our speech-based interface is not an easy task and there are not clear categories of users, we apply fuzzy logic to help us tailor the interface to the specific needs of each user.

The advantage of the proposed system is getting an intelligent mechanism able to adapt dynamically the voice interaction systems to different users and to a user in every moment, an aspect important in very changing environments like the driving. We will integrate the proposed system in the Vitruvius mobile client. This client will be able to define applications using an adaptive voice interface while driving. As a result in many cases, the users will be able to reduce the time used and the number of errors made while defining sensor tracking applications.

The structure of this paper is as follows: Sect. 2 presents the background on adaptive and in vehicle speech recognition systems. In Sect. 3, we describe the Vitruvius platform that is used as the solution. Section 4 consists of a system evaluation. Finally, Sect. 5 covers conclusions and future work.

2 Background

Over the years, vehicles have been equipped with technology that has been turning them into real computers. However, driver distraction interacting with technologies is one common cause of traffic accidents (Neale et al. 2005). Enabling safe and easy interaction with technologies in vehicles by

reducing the cognitive load imposed by the interaction and minimizing head-down time is a crucial aspect to improve the comfort of drivers (Larsson et al. 2013).

For example, Beattie et al. (2014) proposed that spatialized auditory feedback could be used to enhance driver awareness to the intended actions of autonomous vehicles. Papakostopoulos and Marmaras (2012) stated that there is a need for redesigning the conventional display units in vehicles toward a more simple appearance. As is the case with our proposal, authors detected that more experienced drivers have different needs than less experienced drivers. One way to enhance the user experience is through the use of in-vehicle contextual augmented reality that has the potential to provide visual feedback according to the context (Sridhar and Ng-Thow-Hing 2012).

However, it seems that a speech-based interface could serve as an alternative to visual interfaces since drivers would not need to look to the screen or directly manipulate it. Thus, Lee et al. (2001) stated that for working with an email system, there is a 30 % increase in reaction time when a speech-based system is used.

2.1 Speech recognition in vehicles

Speech recognition is a classical research topic, which with the rise of the Internet and mobile devices has experienced a great advance. Nevertheless, even when the speech recognition in controlled situations has achieved a very high level of performance, such performance levels degrade significantly when there is environment noise, specifically between 30 and 100 % accuracy in cars at just 90 mph (Gong 1995). That, together with the applications that can be specifically developed for the domain of vehicles, causes that many scientific works and patents have been arisen. In the same line, many works have been published such as Planet and Iriondo (2012), working on how to detect affective states of people interacting with technology, which can be a hint to know when users are not happy with the application they are using and, therefore, require an adaptation.

One of the most interesting uses for applications in vehicles is to obtain context information. For example in Guo et al. (2015), data is obtained for driver or passengers. Thus, for example, Takahashi et al. (2012) proposed a simple and interesting speech interface with which a user can retrieve vehicle information through a dialogue, so if the user says a keyword that the system recognizes, it returns related information.

Wang et al. (2015) focused on personal smart devices such as smartphones and tablets for navigation applications when users search the destination using the speech interface. Modern devices have had very few errors in the speech recognition when users hold the device on hand. However, high-end devices still have many speech recognition errors in a hands-free mode when the vocabulary that can be used is large.

There are many applications in which speech user interfaces are or will be used since they do not require the user to use her hands or to look at it. It improves safety and provides a quick response in simple tasks that can be asked with one word or a small sentence, which may be of interest to anyone and especially those that may require some action when the vehicle is moving, such as the police Rajamäki et al. (2014). Notwithstanding, other types or user interfaces exist for vehicles such as the more traditional touchscreens or even gesture-based interfaces for vehicle menu navigation (May et al. 2014), but they do not provide the same expressiveness and mobility as the speech user interface (Yankelovich and Lai 1998).

2.2 Adaptive speech user interfaces

Adaptive user interfaces are interfaces that can be adapted to the environment, in which the layout changes according to the needs of each particular user. They focus on showing only relevant information, creating then less confusion to not experienced users and providing speed to more experienced users. As a counterpart, they must be designed with several levels of implementation in mind and designers must have a great knowledge about different user profiles and changes that may occur in their context (Lavie and Meyer 2010).

Regarding speech user interfaces, there is some work such as Gorniak and Roy (2003), who worked on augmenting any Java application with an adaptive speech user interface. To train the system, it constantly searches for correlations between what the user says and what the user does. After the training, speech commands could replace mouse clicks.

SHACER Kaiser (2005) is a speech and handwriting recognizer for capturing out-of-vocabulary terms, dynamically joining them together in the system with the aim of improving subsequent tracking and recognition. It was tested for creating whiteboard schedule charts.

Effective communication with a mobile robot using a speech-based interface is not an easy task. To deal with it, Martinson and Brock (2007) proposed a system in which the robot predicts how the human recognition could be affected by different sound sources (e.g., fans, computers, alarms, vehicles). The novelty of this work is that what it is adapted for is not the recognition performed by the robot, but the sounds it makes to be understood by humans.

There are also some speech-based programming languages. For example, Begel and Graham (2006) presented a speech-based interface for Java programmers, focusing on the lexical, syntactical and semantical ambiguities that do not exist in written source code. The results showed that the learning curve was very low. Experts found programming by

voice slower than typing, but this can be a good strategy in environments when is not possible to type.

In addition, there are more related works such as Arnold et al. (2000), with the idea of proving a way to program without typing for avoiding repetitive stress injuries, or Leopold and Ambler (1997), who investigated a program development interface which responds to the voice, handwriting and gesture in addition to a visual programming environment focusing on improving productivity. However, none of the studied interfaces intended to program seem to have adaptive features such as those presented in this work. These would be interesting to adapt for different users and contexts.

2.3 Fuzzy logic for adaptive interfaces

Adapting an interface to different user profiles and the specific context is not a trivial issue. That is the reason why the use of artificial intelligence is important to address this problem (Langley 1997). For example, artificial neural networks can be used to infer the distraction level of a driver and adjust the interface depending on the context, or fuzzy logic could be used to give accurate results as an inference mechanisms for in-vehicle communication systems (Tchankue et al. 2011).

Mäntyjärvi and Seppänen (2003) used fuzzy context information to adapt context-aware applications for handheld devices. There are many other studies such as Soui et al. (2013) that presented a work to personalize user interfaces using fuzzy logic. Their work is based on several relations to select which interface components are useful depending on the context.

However, in our work we performed a somewhat different approach in which fuzzy logic is used to adapt how systems communicate with users in a speech-based interface that can be seen as a small domain-specific programming language to generate applications in the domain of vehicles.

3 Platform description

The Vitruvius platform combines several technologies to achieve pioneering functionalities that other solutions are missing. The main objective of the platform is that non-expert users or drivers can make real time consults with several complexity levels based on geographical information and technical data from one or more vehicles (Fernandez et al. 2013). The characteristics involving this process require the consultations to be made through a speech recognizing system to maintain driver's attention on the road. Queries are analyzed and processed by the application core. The core combines an interpreter and a dynamic web application generator that uses information from a large database that manages real time and past information from vehicles. The generated web application is created based on the instruction of the query to obtain data in real time from vehicles included in the query. The resultant information is displayed according to the user demands. For most, the results must be presented quickly since the system is aimed for user sudden necessities.

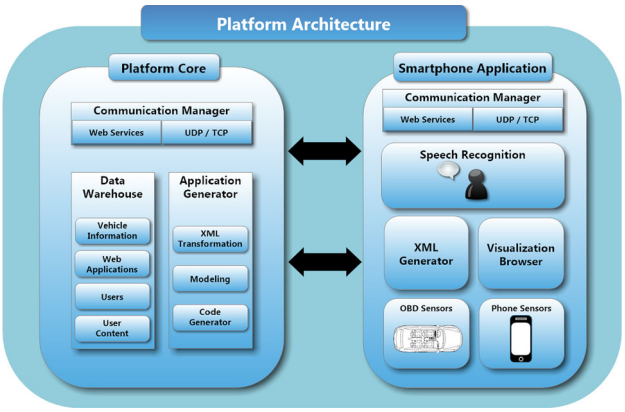

Fig. 1 Platform's architect diagram

On Vitruvius platform, users can perform a double role: (1) all users can make voice queries that will generate multimedia web applications, and (2) users send sensor data from their vehicles to the platform so they can be used in the generation of applications. Data can be shared globally or with the specified users.

From a technical point of view, the platform follows a client–server structure (Fig. 1). The server works as the core of the platform being in charge of the user's queries, management of vehicle data and the generation of web multimedia applications. Client applications can be used in two ways: the simplest consists of a mobile application that users can use to create consults with a voice interface and the complex fragment is reserved for driving users that upload their vehicle sensor data to other users. Collecting data from the vehicle requires the installation of a hardware OBDII.

Below the more relevant parts of the platform architecture will be detailed.

3.1 Platform core

The platform's core is located in the server side. It has three main functionalities: (1) Manage the reception and delivery of the data that the platform needs. The information consists of technical data from vehicles connected to the platform that will allow an online access to the generated applications, (2) Management and storage of data received from vehicles connected to the platform, and (3) Processing the user specifications to create a base for the web multimedia application generation on-demand. Applications perform data analysis stored in the platform to satisfy the user needs.

The platform core contains specific modules in charge of the functionalities presented above:

1. *Communication manager module* To manage the constant fluid of data from client applications, different kinds of communication protocols have been used. The information received from the vehicle and mobile device sensors includes speed, acceleration forces, geographical points, engine revolutions per minute, coolant temperature, throttle position, etc.…
2. *Data warehouse module* All the information regarding the vehicle information and user preferences is contained in the warehouse. Additionally, a web application server is running to ensure the availability for the deployed applications of users. This information can be added through the web interface since adding this information through the voice interface can be very tedious. The voice interface will look for the user's content when creating new applications.
3. *Application generator module* Every user of the application can make its own web applications through the application generator. A modeling XML domain-specific language has been created to generate the applications. The XML file is built by the speech recognition system on the mobile device application that generates a tree with the user input option that will later convert into an XML file. The XML is written in an extensive specific domain language, especially designed for the platform that allows the specification of vehicle-related programs. When the XML file arrives to the platform core, it is processed using an interpreter. The interpreter is in charge of building a memory tree that helps generate the code for each of the elements found. Once the code is finished, it is deployed in the web application server in the Data Warehouse module, allowing it to be accessed in a few seconds after the creation of the application.

3.2 Client applications

Client applications are an essential part of the platform. Clients can interact with the platform in three non-exclusive ways. (1) *By sharing information from their vehicle* connecting OBDII hardware to the vehicle that communicates to a Smartphone, tablet, or laptop through a Bluetooth interface (Espada et al. 2013). (2) *Generating applications* The user uses a voice assistant from his mobile device to determine the web application that he wants to develop. Once the application specification has ended, the information is sent the platform's core to create the application. (3) *Application usage*, the applications that are generated with HTML5, which is based on web standards, implies that they can be visualized from any standard Internet browser.

The client applications are used to obtain the information from vehicles and to create and display applications. Although the application is able to upload data and generated applications at the same time, the application can be set to only upload data or to just create applications.

3.2.1 Voice generated applications

The system is designed to be operated in moving vehicles. The driver should be able to use the system while operating the vehicle without looking to a device. To achieve this, we have created a voice interface for the application. The voice interface allows a driver to create an application without having to take the eyes off of the road. For example, a father while driving his car would be able to create an application that will notify him when his son has exceeded a certain speed or RPM in his car.

The application interacts with the user through conversational interactions. The voice interface when activated by the user follows the structure shown in Fig. 2. Once the application detects a new application command, it will start questioning through all the possible options. The application will ask about the following specific items requiring short answers.

The process definition flow of the application can be divided in the following stages:

1. What vehicle or vehicles it should select? Vehicles can form groups so they can be easily selected in the application.

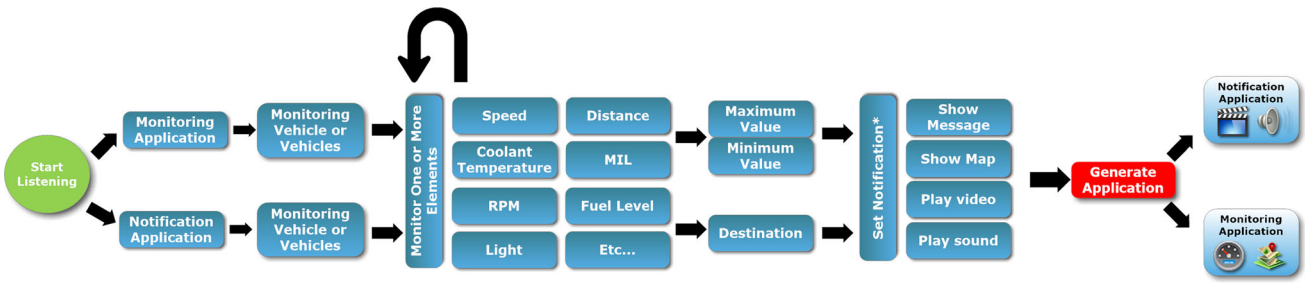

Fig. 2 Voice interface generating flow. Application generating process with initial state, processing nodes and the resulting possible applications

2. What vehicle parameters should the application monitor? All the available parameters from the mobile device and the vehicles OBDII interface can be selected to generate the new application.
3. When should parameters trigger the notification? Parameters can either exceed a certain value or be under a certain value to trigger the notifications. Additionally, several limits can be selected with the same parameter. For example, a vehicle can trigger a notification when its speed is 0 km/h and when the speed is over 120 km/h.
4. What sort of notification or monitoring should the application show? The application allows specifying a constant monitoring of the state of one or more elements, but also allows establishing a series of sound signals when some of the elements enter a determined range, for example, a low fuel level.

The application translates the tree into an XML-specific domain language built for the platforms application generator. The XML is sent through web services to the core server.

3.2.2 Visual generated applications: monitor/notification system

The generated applications are aimed to be visualized on any conventional Internet browser. This is why the visual system included in the generated web applications has been implemented using web standards and conventional web languages (HTML, CSS, JavaScript, etc.). The mobile device application uses a regular attached Internet browser to visualize the applications. The application launches itself on an internal browser in a few seconds after the user specifies the application. The application has two behaviors (1) *Monitoring* they constantly show one or more parameters and (2) *Notifications* executing an explicit action when a parameter condition has been reached. The behaviors can be combined, for instance, an application could monitor an element and then create a notification when a condition is reached.

Monitoring The displayed elements on the applications depend on the parameters specified by the user. Once a new application has been created, a map with the monitored vehicles is shown displaying each of the desired parameters. Visual indicators can monitor parameters, such as engine RPM, CO_2 emissions or fuel consumption.

Notification Once a notification is triggered, one of the following notifications can be displayed.

- Display a message. The application can display a message to the user.
- Play a sound. Sound can be added while displaying the message.
- Play a video. Video can be also displayed while displaying the message.
- Show a map. The application can open a map that will monitor the vehicle or vehicles that are being tracked showing the parameters marked with pointers in a map.

Since applications can be visualized in many devices, they are easier to share. Depending on the permissions used to share the applications, one same application could be visualized by a group of users, such as a vehicle fleet. Additionally, users that are not in vehicles can also use it. For example, it could be used to get ready for the arrival of a vehicle or to check the traffic information in an area.

Vehicle users would be able to generate applications while driving with this platform. With only a smartphone, they can interact with other users of the road. We have developed some prototypes that use the proposed platform. Then, the main objective of the prototypes is to show the application definition process from users creating real applications that can be made with the system.

The mobile device application has two different modes that work independently. Upload only mode obtains sensor readings from the vehicle and the mobile device and uploads them to the system core. So, in order to monitor a vehicle, a smartphone with the running application and Internet connection should be mounted in the vehicle and the uploaded mode activated. Adding the device with the Bluetooth OBDII adapter is optional. If the adapter is not paired, it will just upload the information from the device inbuilt sensors. Every sensor has its privacy settings and can be modified in the application.

To create a new application, the generator mode should be used. Both modes are independent and can be used simultaneously to create and upload sensor information. The voice interface system guides users and allows them to interact with the application through the speech recognition system. The application communicates all the actions through speech and waits for the user's response. The interface will recognize key words and will start building a consulting graph that will be converted into the application. Once the application is created, it will start running and it will be displayed to the user.

3.3 Intelligent voice interaction system

Voice interactions must be adapted to the user's abilities to obtain a good user experience. This objective is not always easy to achieve. Some systems are designed for a wide range of users and it is very difficult to determine all the possible user profiles. In other situations, user profiles can be determined, but they are difficult to match to the real users in every situation since the same user does not necessarily retain

always the same skills. In some environments, the skills and the ability of a user can be affected by several external factors. In the Vitruvius platform, we have to deal with both problems. We designed our voice interaction system to work on a wide range of users with different experience and abilities. In addition, the same user can interact in different ways depending on the external factors, making their skills and concentration not always the same. In summary, the Vitruvius Voice Interaction system must be capable of adapting its functionality to users with different skills in concrete moments because the skills of a concrete user may vary at any time (for instance during driving) due to external factors.

When a user interacts with a voice application, the user has to understand what the application is doing and what the user can do in every moment. Common voice interaction mechanisms use messages to describe current activities and the state of the application. User commands are used to send information or orders to the application. Not all the users have the same skills. Most voice interaction applications make an initial analysis of the potential users. Subsequently, the application classifies users in generic groups and creates stable profiles for these users with messages and commands that are easy to understand. Some voice interaction applications define a unique profile; in this case, some users could have more difficulties to understand the application. Other applications include different profiles allowing selection of the most suitable for different skills and experience. In most cases, the profiles are categorized based on the user experience and abilities. A user with a lot of experience is able to understand the current state of the application with a short message, and can manage the application with complex commands. On the other hand, a user with limited experience or a user surrounded by distractions needs long messages with a lot of information to understand the application and commands sets.

The approach of the intelligent Voice Interaction Systems consists of the design of three initial profiles (beginner, intermediate, and advanced). The profiles are set to go from easy to use to hard. The beginner profile contains very long descriptions. The options of the users are very restricted. However, in the advance profile, the description is short and only contains the minimum necessary data making users free to execute complex commands. The three profiles allow users to generate the same applications but not all apply the same level of guidance or assistance. It is easier to make applications with the beginner profile but the user needs a lot of time to complete an action. The same action could be completed faster using a complex command in the advance profile. This makes all the profiles adequate for a concrete type of user. In the case that a profile is used in a non-appropriate user, the user experience could be very bad; it could lead to make a lot of errors or spend too much time completing action that the users already master.

Probably, some users can fit directly in one of these three profiles, but we aim to design an application that is valid and efficient for a lot of different user profiles. Some users could understand well some parts and experience problems in others. Our approach is to generate an intelligent voice interaction system able to detect the user needs in every monument and adapt the indentation to the adequate profile in every monument; therefore, we could solve the current challenges of the voice interaction system in the platform by covering a wide range of different profiles and adapt the system to the users abilities which can be change dynamically due to external factors during the driving.

An approach we considered for our voice system included simplifying the functionality of the voice interaction system. Less commands normally mean that the application will differentiate between actions better. Moreover, fewer commands allow users to memorize all the keywords to use the full potential of the application.

Another option that we took into consideration was to increase the verbal description that the system gives to users. This system worked great with new users. Users gain experience and practice with the voice interface. The problem comes when too much detailed instructions are narrated. In complex systems, it can take too much time and frustrate users. Instructions could take significantly more time. This would make users want to use an alternative non-voice classic interface, where their perception of speed due to creating action is superior.

The final approach we took was to add an extra layer of intelligence to the voice interaction system. Figure 3 shows the conceptual diagram that we introduced to create a hybrid approach from both previously described approaches. Ideally depending on the level of the user, we could be more descriptive or just provide a quick experience. We established 3 levels of users: beginner, intermediate and advanced.

Beginner The application becomes very descriptive. Every time the user interacts with the device, feedback is obtained from the actions of the user. Recognition system can handle long and short answers.

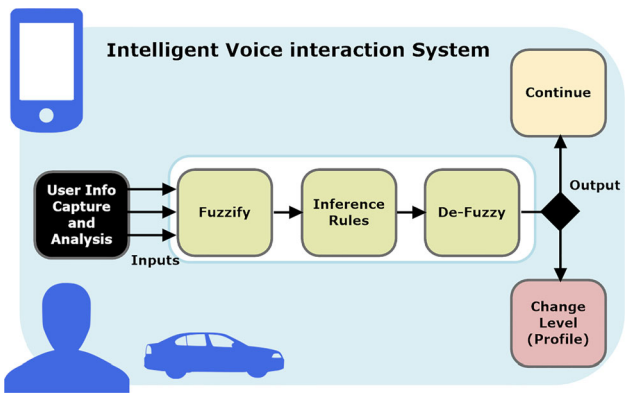

Fig. 3 Conceptual diagram of the intelligent voice interaction system

Intermediate The intermediate level is the level by default when you first start the application. It helps the user to understand the flow of the application. It is descriptive enough and it asks for general items. It does narrate all the options possible.

Advanced Expert level has been design for the user that has mastered the creation of apps. The description with instructions for the user has been minimized to the maximum. This helps keep the time to create an application to the bare minimum. An advanced user could create a complex application in under a minute.

To define an intelligent voice interaction system able to modify the level dynamically we created a fuzzy membership function. We analyze data from the user activity, the voice interaction system is continually extracting info about the user activity and with a fuzzy analysis of these parameters the application can know if the user is comfortable with the current level (beginner, intermediate or advance). If the user is doing well, the system will maintain the user in the same level, but if the user has problems understanding or if he is a confused about the commands the system will change the level. Likewise, if the user is very accurate and fast, the system could put a more advance level to try to save time to the user, and improve the user experience.

The intelligent voice interaction system is continually capturing ten different parameters related to the user activity.

The use of "Repeat" command This parameter counts the number of times that the user uses the repeat command. This command will repeat the last speech from the application.

The use of "Help" command This parameter counts the number of times that the user uses the help command. The help command will provide additional information to the user. It will explain what kind of information and keywords that the user can use. It also provides tips and suggestions for a better understanding of the app.

The use of the "Go back" command cancels the current selection and goes back to the previous question asked.

The use of the "Start Over" command completely cancels the creation of the application. The user will be prompted to the main screen of the voice recognition application.

Applications successfully created Every time an app is created the user will gain more experience on the system. This is recorded and acknowledged in our system.

Actions completed The number of commands completed is measured by the system to identify successful actions with the user.

Actions not completed Every time a user does not complete an action it is tracked by the system.

Use of frustrated vocabulary The application contains a list of offensive words such as swear words. Every time the user interacts with the system the application checks if the users uses the vocabulary in the list. We consider frustration of the user can be measured when this kind of vocabulary is used.

The use of keywords out of context The application registers every time the user uses a keyword out of context. This denotes that the user is disoriented or that is not familiarized with the system.

Average number of words used We also count average number of words used. Advanced users tend to realize that the system works understanding keywords and use them precisely. Inexperienced users can be inclined to use large speeches and giving extra information not needed.

Based on some set of tests with real non-expert users and expert users, we determined for every parameter which values were optimal to interpret the user understanding. The selected parameters are very useful to see if the user is having a correct interaction with the application. At this point, we defined a set of fuzzy logic parameters. We used the library jFuzzy logic (Cingolani and Alcala-Fdez 2012; Cingolani and Alcalá-Fdez 2013), to implement the fuzzy log in the voice interaction System, which is an Android Application.

The first step was to define the ranges for each one of the 10 parameters captured. The fuzzification that we generated based on the data that we have collected and refined is shown in Fig. 4. The fuzzy membership functions combine the use of trapezoidal and triangular functions.

Additionally in the future, we can consider including new parameters such as voice level changes and background noise. These new parameters would help identify more scenarios where extended commands would be difficult to use. For example if the user is trying to use the application with a noisy background and starts raising its voice, the application should recognize this and reduce the level of the application; consequently, the user will only require concise commands instead of elaborated speeches.

Based on the test performed with non-expert users and expert users, we defined a set of fuzzy rules to determine which of the three levels implemented is more appropriate for the user. To obtain the most appropriate level of the voice interaction system, we defined an output variable "user_level". The value of the variable is used by the intelligent voice interaction system to determine the level that has to be used. The application is able to change the level in every question. The variable "user_level" can take the three values: beginner, intermediate, and advanced (Fig. 5).

```
DEFUZZIFY user_level
    TERM beginner := (0,1) (2,1) (4,0);
    TERM intermedium := (3,0) (5,1) (7,0);
    TERM advanced := (6,0) (7,1) (10,1);
    METHOD : COG;
    DEFAULT := 5;
END_DEFUZZIFY
```

The intelligent voice interaction system will change the current level based on the result of the value. At the end of

the process, after the experts advice and data collected in the scenarios with non-expert users, the set of fuzzy rules selected is as follows:

```
RULEBLOCK No1
AND : MIN;  /* Use 'min' for 'and'
(also implicit use 'max' for 'or' to
fulfill DeMorgan's Law)*/
ACT : MIN;  /* Use 'min'
activation method*/
ACCU : MAX; /*  Use 'max'
accumulation method*/

RULE 1 :
IF appsCreated
    IS a_many AND RepeatCount
    IS a_few AND HelpCount
    IS a_few
THEN user_level IS ad-vanced;

RULE 2 :
IF appsCreated
    IS a_enough AND RepeatCount
    IS a_few AND HelpCount
    IS a_few
THEN user_level IS intermedium;


RULE 3 :
    IF RepeatCount
    IS a_enough OR HelpCount
    IS a_enough
THEN user_level IS beginer;


RULE 4 :
IF SwearWordCount
    IS a_few OR StartOverCount
    IS a_enough OR BackCount
    IS a_many OR KeyWordsOutOfContextCount
    IS a_many OR UserNotCompletedActionsCount
    IS a_many
THEN user_level IS beginer;

RULE 5 :
IF RepeatCount
    IS a_enough AND HelpCount
    IS a_enough
THEN user_level IS beginer;

RULE 6 :
IF appsCreated
    IS a_enough AND StartOverCount
    IS a_few AND KeyWordsOutOfContextCount
    IS a_few AND SwearWordCount
    IS no AND HelpCount
    IS a_few AND completedActions
    IS a_enough
THEN user_level IS intermedium;

RULE 7 :
IF averageCountWords
    IS a_few AND RepeatCount
    IS a_few AND HelpCount
    IS a_few AND completedActions
    IS a_many
THEN user_level IS advanced;
```

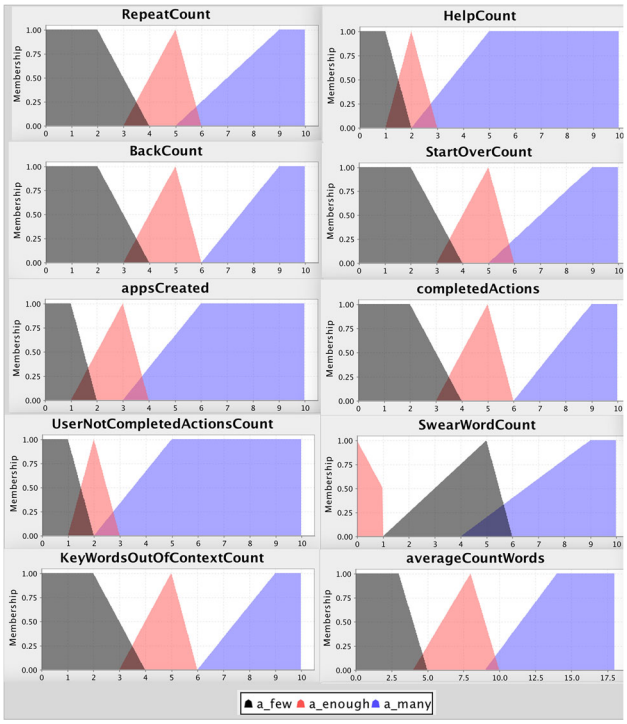

Fig. 4 Fuzzy functions for the ten input variables

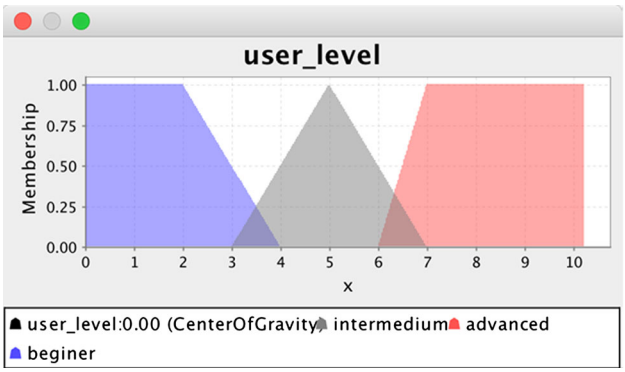

Fig. 5 Fuzzify functions for the output variable

These rules are able to calculate what of the three levels is more appropriate for the user in every moment based on his current activity.

4 Evaluation

The evaluation process of the proposal is formed by two different analyses. In the first part of the evaluation, users generated two different applications using the voice interface. No visual help or guidance was provided. During this part of the evaluation, we wanted to compare the time that users took to define an application and the numbers of errors made. For this purpose, we selected a group of seven users. They

had to define two Vitruvius applications using the standard voice client and the proposed fuzzy client with adaptable functionality in every user interaction.

In the second evaluation, we analyze the optimum time needed to define two applications with the three different profiles that are included in the proposal (beginner, intermediate and advanced). With this information, we made a simulation emulating users and estimating the time that users could save on the definition of an application if the voice client would move to a higher level when defining the application.

The scenarios were designed to represent two common use cases of the platform. In these scenarios, the user defines a petition of information related to a vehicle. The petition generates a specific application that meets the user requirements. The two selected scenarios are described below:

Application 1. Parental Control Scenario

A controlling parent who has just found out that his son (that has just got his license issued) is going on a long trip in his beloved car for the first time. The parent is driving his own car to work and is still not comfortable with his son's driving abilities. So the parent feels the urge to monitor his son's trip to feel calmer and to ensure that his son's driving is adequate in his car. Additionally, he would like to get notified when his son arrives to his final destination.

So he decides to create a new application to monitor his son. The application will notify him when his son arrives to the final destination, if he exceeds 120 km/h, or if the engine exceeds 4000 RPM.

The following conversation is a fragment of an example of how the application could be generated; bold words are the key words that the application recognizes:

1. Voice application—What can I do for you?
2. User—I want to create a ***new notification*** application.
3. Voice application—What vehicle or vehicles do you want to monitor?
4. User—***Car 7***
5. Voice application—What parameter do you want to monitor?
6. User—I want to monitor the ***speed*** of the car.
7. Voice application—What speed should be exceeded to notify you?
8. User—***120***
9. Voice application—What speed minimum speed to notify you?
10. User—***Don't*** set minimum speed.
11. Voice application—Do you want to monitor more parameters?

…

…

20 User—***No*** more parameters

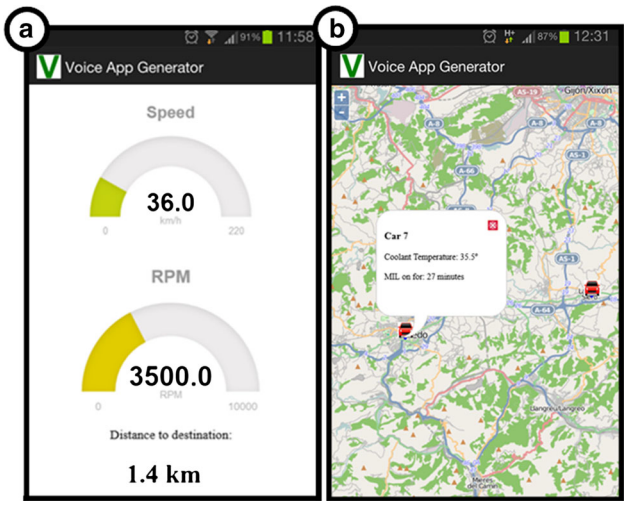

Fig. 6 At the left, prototype 1 application is monitoring the "Car 7" vehicle. At the right, prototype 3 application is displaying in a map information about vehicle 7

21 Voice application—Generating and displaying application. Wait a moment please...

The application then will generate and display the new web application. The application will show speed and RPM gauges with the "Car 7" parameter values and the distance left to arrive at the final destination (Fig. 6). Whenever any of the parameters previously set exceed the values, the application will inform with a message to the user at the same time that it is playing a beeping sound.

Application 2. Track a vehicle

The application tracks a vehicle and displays a map with the location of the vehicle and the desired parameters.

1. Voice application—What can I do for you?
2. User—I want to ***track a vehicle***.
3. Voice application—What vehicle or vehicles do you want to monitor?
4. User— ***Truck 23***
5. Voice application—What parameter do you want to monitor?
6. User—I want to monitor the ***fuel tank*** *percentage*
7. Voice application—Do you want to monitor more parameters?
8. User— ***NO***
9. Voice application—Generating and displaying application. Wait a moment please...

The application will create and display a new web application with a map that will show the vehicles that are being tracked.

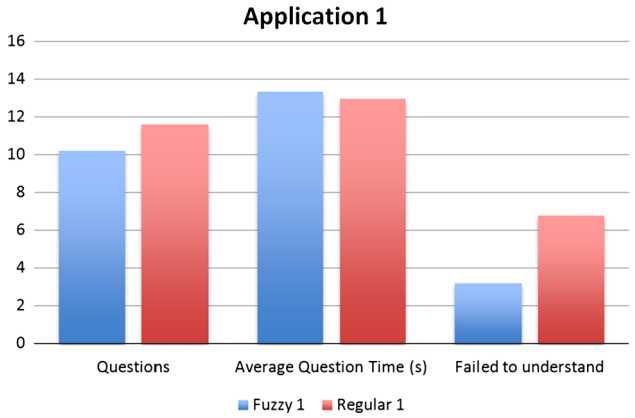

Fig. 7 Average data collected in the definition of the application 1. Average number of questions, average question time (in seconds), and average number of failed interactions

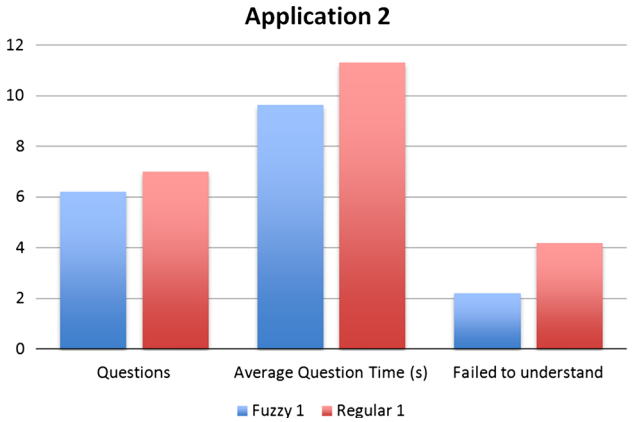

Fig. 8 Average data collected in the definition of the application 1. Average number of questions, average question time (in seconds), and average number of failed interactions

When pressing on a specific vehicle, it will show the fuel tank percentage and the coolant temperature level (Fig. 6b).

Evaluation 1 results

During the user interactions, we used a log system to monitor their response times, errors and the commands that they said. Figure 7 shows the average number of questions that the users needed to reply, the average question time (in seconds), and the number of failed interactions that users had to create application 1.

Based on the result, we can suggest that the Fuzzy Voice client was useful at reducing the average number of questions that users needed to define the application. The number of questions was reduced from 11.3 to 10.2, *a reduction of more than 9 %*. This reduction means that users had a better comprehension of the questions and needed fewer questions to define the application. To endorse the improvement on question comprehension, we also observed that the average number of failed interactions was reduced from to 6.8 to 3.2, *a reduction of more than 50 %*. In the obtained result, we can also observe that the average questions that users needed to complete the application were increased from 12.9 to 13.3, *a increase of less than 4 %*. This is due to questions in lower levels including longer descriptions and hints.

The chart Fig. 8 shows the same data as we showed for the first application but in this case for application 2. The main difference between both applications is that application 2 requires monitoring less parameters than application 1. Not many questions are needed to generate these applications. As the results indicate that the process did not require many questions, consequently the client did not have many chances to obtain user data and adapt the voice interpreter to a different profile.

In the case of application 2, we can see a set of results in the same trend as in the application 1. However, since application 2 requires less commands than the application 1, all

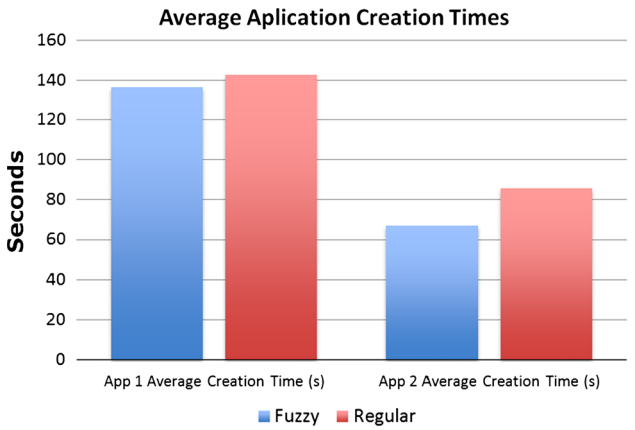

Fig. 9 Average time that users needed to define both applications

the data have significantly lower values. The average number of questions that users needed to define the application was reduced from 7 to 6.2, *a reduction of more than 10 %*. The average number of failed interactions was reduced from to 4.4 to 2 *a reduction of more than 54 %*. In this case, also the average time to respond to the questions was reduced from 11.3 to 9.6, *a reduction of more than 15 %*. However, this value must be interpreted with caution, since the average time to reply to questions in an application with a low number of questions can easily be altered.

In Fig. 9, we show the average time that users needed to define both applications.

We observe in the results that the time that users needed to define the application was reduced in both cases; about 5 % in the application 1 and 20 % in the application 2. Although in this evaluation process, the proposal can be slightly useful to reduce the time that the users need to create applications. There is a notable efficiency improvement in reduction of the numbers of errors that users made; 50 % in application 1 and 54 % in application 2. During the definition of an application process, the proposal is able to get two benefits: dynamic

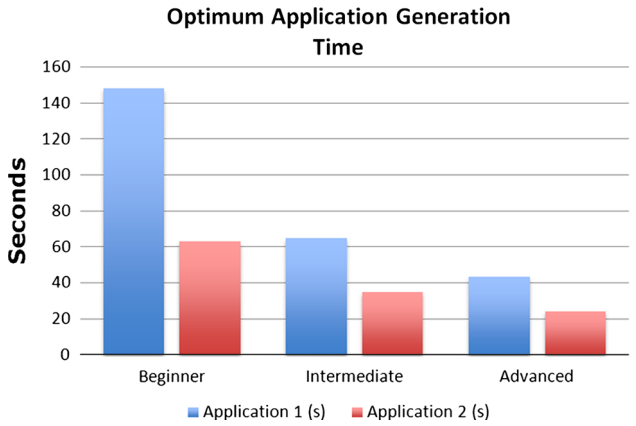

Fig. 10 Optimum time obtained by the expert in the generation of the two applications in the three different levels (*beginner*, *intermediate* and *advanced*)

changes to a higher level to reduce the time that the user needs to define the application, and dynamic changes to the lower levels to reduce the number of user failed interactions. However, the system is increasing the time that some users need to define the application with changes to lower levels.

During the evaluation, we obtained both types of dynamic changes, to higher levels and lower levels. Although judging the results, the dynamic changes to lower levels were more common. The reduction in the user failures is the most significant value. This is something very coherent since users that participated in the evaluation were not experts in the use of the system. There were only a few situations where the client had considered that the current user had enough domain to increase the level.

Evaluation 2 results

To evaluate all the key issues of the proposal, we need expert users to define the capacity to reduce the time that users spend on the definition of the applications. Expert users have a great domain of the client. In this case, chances are that the application considers to increase the level in some part of the process. With a high level profile, users could define the application with fewer questions and less time.

The times obtained with an expert user in the three different modes are shown in the next chart Fig. 10. The times are obtained when the experts defined the 100 % of the application in the three modes; for this evaluation, the intelligent voice client was disabled.

If the intelligence voice client is active, the application can change the level of the voice interpreter. If the user is doing well, the client will increase in real time the level in some questions to reduce the time that the user needs to define the application. For this part, we used the advanced times to make a simulation of different "expert" users. Users start the process in intermediate level, just as in the default client (without intelligence), but in the intelligent client some parts

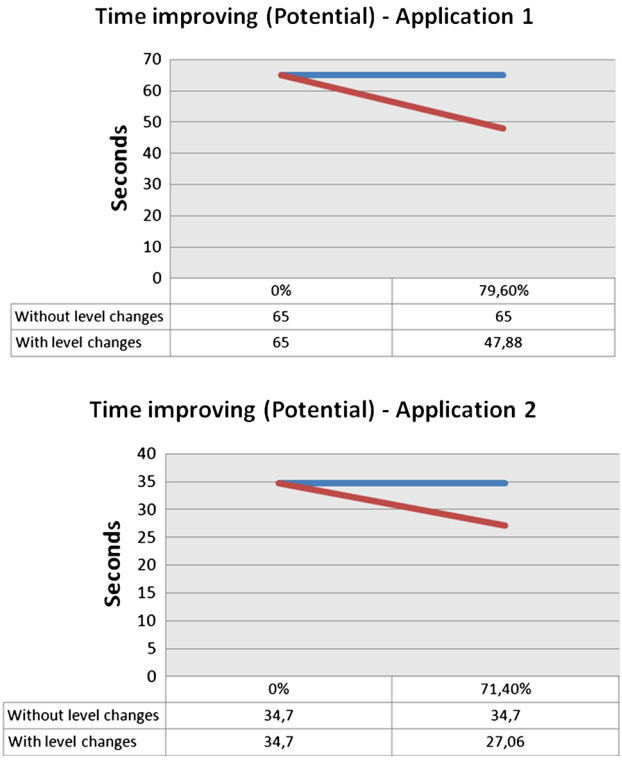

Fig. 11 Potential time improving the combined use of the intermediate and advanced profile versus the standard (intermediate profile), the advanced profile is using 0–79.6 % in the definition of the application 1 and 0–71.4 % in the definition of the application 2

of the process can be done using a higher level. Based on the first part of the evaluation, we estimate that around 0–79.6 % could be done in level different to the start level in the application 1 and 0–71.4 % in the application 2 (Fig. 11). The estimation of the time reduction in the definition process of both applications is shown in the following charts.

Based on the estimates performed, it could be possible in a favorable scenario that level changes could cause an important reduction of the time invested in the definition of an application for an expert user. According to the estimations in the best case, the time could be reduced by more than a 24 % in the case of the application 1 and more than a 22 % in the application 2. Those values are an estimation of the time that could be aimed by a user with high domain of the application in some scenarios. We can observe that the optimum time is very far from the reduction of the time that the users got in the results of the first evaluation. They only get about 5 % of time reduction due to users that did not have a great domain of the voice client and the process to definition was so long. However, in the second application, users got a very good result. It was the second application that they generated, so they had more experience. Additionally, application 2 definition process was shorter than application 1 and there were less opportunities to make mistakes, so in case two real users obtain a reduction of time very close to the optimum.

5 Conclusions

In this paper, we present a novel fuzzy voice interaction approach that has been included in Vitruvius platform clients. The aim is to improve the user experience, reducing the number of failed interactions that non-expert users make during the communication with the voice client. We also tried to reduce the time that users spent in the generating process. The research work focuses on the present challenges of voice interaction systems in very dynamic specific scenarios. Most voice interaction systems are based on one or more profiles. For example: beginner, intermediate, expert profiles modes that provide different information and instructions. The static user profiles are not enough to obtain a good user experience, especially in complex processes with many different parts, or in environments in which the external factors can affect the skills and attention of the users.

The proposed voice interaction system includes three different profiles for voice interaction. These profiles include more or less detail level in the instructions and in the options. The proposed system captures real time information about the user interaction and uses the information to determine if the user is feeling comfortable with the voice assistant. The system uses a set of fuzzy rules to determine if it is necessary to change the current user profile to obtain the best user experience by reducing the number of failed interactions and the time to complete the process.

According to the obtained results in the evaluated scenarios, the system was useful to meet the objectives of the research. In the evaluated scenarios, the proposal allows a reduction in the number of failed interactions of non-expert users in more than 50 % (app1) and more than 54 % (app2). Likewise, the proposal showed to be slightly useful reducing the time that non-expert users need in the process. It was reduced more than 4 % (app1) and more than 20 % (app2). The possibility of a bigger time reduction is greater on users with the great domain of the system defining relatively short processes. In these interactions, users have more possibilities to do a fluid interaction that allows the system to upgrade the profile level. In this use case, we tried the proposed system with the Vitruvius platform as a user case, but the approach can be applicable to many other voice interaction systems that share some of the Vitruvius scenario conditions.

References

Arnold SC, Mark L, Goldthwaite J (2000) Programming by voice, vocal-programming. In: Proceedings of the fourth international ACM conference on assistive technologies, Arlington, USA, pp 149–155

Beattie D, Baillie L, Halvey M, McCall R (2014) What's around the corner? Enhancing driver awareness in autonomous vehicles via in-vehicle spatial auditory displays. In: Proceedings of the 8th nordic conference on human-computer interaction: fun, fast, foundational, vol 8. Helsinki, Finland, pp 189–198

Begel A, Graham SL (2006) An assessment of a speech-based programming environment. In: Proceedings of IEEE symposium on visual languages and human-centric computing, Brighton, UK

Bouchon-Meunier B, Valverde L (1999) A fuzzy approach to analogical reasoning. Soft Comput 3(3):141–147

Cingolani P, Alcalá-Fdez J (2013) jFuzzyLogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming. Int J Comput Intell Syst 6(sup1):61–75

Cingolani P, Alcala-Fdez J (2012) jFuzzyLogic: a robust and flexible fuzzy-logic inference system language implementation. In: Proceedings of IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1–8

Cueva-Fernandez G, Espada JP, García-Díaz V, García CG, Garcia-Fernandez N (2014) Vitruvius: an expert system for vehicle sensor tracking and managing application generation. J Netw Comput Appl 42(1):178–188

Cueva-Fernandez G, Espada JP, García-Díaz V, Gonzalez-Crespo R (2015) Fuzzy decision method to improve the information exchange in a vehicle sensor tracking system. Appl Soft Comput (in press)

Dutta S, Chakraborty MK (2015) Fuzzy relation and fuzzy function over fuzzy sets: a retrospective. Soft Comput 19(1):99–112

Espada JP, Díaz VG, Crespo RG, Martínez OS, G-Bustelo BCP, Lovelle JMC (2013) Using extended web technologies to develop bluetooth multi-platform mobile applications for interact with smart things.Inf Fusion 21(1):30–41

Fernandez GC, Espada JP, Díaz VG, Rodríguez MG (2013) Kuruma: the vehicle automatic data capture for urban computing collaborative systems. Int J Interact Multimed Artif Intell 2(2):28–32

Ghafoor Y, Huang Y-P, Liu S-I (2015) An intelligent approach to discovering common symptoms among depressed patients. Soft Comput 19(4):819–827

Gong Y (1995) Speech recognition in noisy environments: a survey. Speech Commun 16(3):261–291

Gorniak P, Roy D (2003) Augmenting user interfaces with adaptive speech commands. In: Proceedings of the 5th international conference on multimodal interfaces, Vancouver, Canada, pp 176–179

Guo L, Ma J, Chen Z, Zhong H (2015) Learning to recommend with social contextual information from implicit feedback. Soft Comput 19(5):1351–1362

Kaiser EC (2005) Shacer: a speech and handwriting recognizer. In: Proceedings of the international conference on multimodal interfaces (ICMI), Trento, Italy, 2005, pp 63–70

Kóczy LT (2006) Fuzziness and computational intelligence: dealing with complexity and accuracy. Soft Comput 10(2):178–184

Langley P (1997) Machine learning for adaptive user interfaces. In: Proceedings of the 21st annual German conference on artificial intelligence: advances in artificial intelligence, Freiburg, Germany, pp 53–62

Larsson S, Berlin S, Eliasson A, Mecel AB, Kronlid F, Talkamatic AB (2013) Visual distraction test setup for an multimodal in-vehicle dialogue system. In: Proceedings of The 17th workshop on the semantics and pragmatics of dialogue, Amsterdam, Netherlands, pp 215–217

Lavie T, Meyer J (2010) Benefits and costs of adaptive user interfaces. Int J Hum Comput Stud 68(8):508–524

Lee JD, Caven B, Haake S, Brown TL (2001) Speech-based interaction with in-vehicle computers: the effect of speech-based e-mail on drivers' attention to the roadway. Hum Factors J Hum Factors Ergon Soc 43(4):631–640

Leopold JL, Ambler AL (1997) Keyboardless visual programming using voice, handwriting, and gesture. In: Proceedings of the 1997

IEEE symposium on visual languages (VL '97), Capri, Italy, pp 28–35

Lledó LD, Bertomeu A, Díez J, Badesa FJ, Morales R, Sabater JM, Garcia-Aracil N (2015) Auto-adaptative robot-aided therapy based in 3D virtual tasks controlled by a supervised and dynamic neuro-fuzzy system. Int J Artif Intell Interact Multimed 3(2):63–68

Mäntyjärvi J, Seppänen T (2003) Adapting applications in hand-held devices using fuzzy context information. Interact Comput 15(4):521–538

Martinson E, Brock D (2007) Improving human-robot interaction through adaptation to the auditory scene. In: Proceedings of the ACM/IEEE international conference on human-robot interaction, Arlington, USA, 2007, pp 113–120

May KR, Gable TM, Walker BN (2014) A multimodal air gesture interface for in vehicle menu navigation. In: Proceedings of the 6th international conference on automotive user interfaces and interactive vehicular applications, Seattle, USA, pp 1–6

Neale VL, Dingus TA, Klauer SG, Sudweeks J, Goodman M (2005) An overview of the 100-car naturalistic study and findings. Natl Highw Traffic Saf Adm 1(05–0400):1–10

Papakostopoulos V, Marmaras N (2012) Conventional vehicle display panels: the drivers' operative images and direc-tions for their redesign. Appl Ergon 43(5):821–828

Planet S, Iriondo I (2012) Comparative study on feature selection and fusion schemes for emotion recog-nition from speech. Int J Interact Multimed Artif Intell 1(6):44–51

Rajamäki J, Timonen T, Nevalainen J, Uusipaaval-niemi H, Töyrylä T, Arte E (2014) Human-machine interactions in future police vehicles: applying speech user interface and RFID technology. Int J Syst Appl Eng Dev 8(1):163–170

Silva W, Serra G (2014) Intelligent genetic fuzzy inference system for speech recognition: an ap-proach from low order feature based on discrete cosine transform. J Control Autom Electr Syst 25(6):689–698

Soui M, Abed M, Ghedira K (2013) Fuzzy logic approach for adaptive systems design. In: Proceedings of interaction human-computer, towards intelligent and implicit interaction, Springer, Heidelberg pp 141–150

Sridhar S, Ng-Thow-Hing V (2012) Generation of virtual display surfaces for in-vehicle contextual augmented reality. In: Proceedings of IEEE international symposium on mixed and augmented reality (ISMAR), Japan, Fukuoka

Takahashi J, Katae N, Harada S, Matsumoto C, Noguchi Y, Murase K, Watanabe K, Matsuo N, Iwamida H, Fukuoka T (2012) Intuitive speech interface for vehicle information systems. In: Proceedings of 19th ITS world congress. Austria, Vienna

Tchankue P, Wesson J, Vogts D (2011) The impact of an adaptive user interface on reducing driver distraction. In: Proceedings of the 3rd international conference on automotive user inter-faces and interactive vehicular applications, Salzburg, Austria, pp 87–94

Wang Y, Zhu J, Zheng T, Gao F, Guo X (2015) Comparing three smart device setups for the use of speech interface in desti-nation search while driving. In: Proceedings of transportation research board 94th annual meeting, Washington, USA, no 15–0469, pp 1–11

Yankelovich N, Lai J (1998) Designing speech user interfaces. In: Proceedings of CHI 98 conference summary on human factors in computing systems, Los Angeles, USA

Zadeh LA (1965) Fuzzy sets. Inf Control 8(3):338–353